

TRACKING AND PLANNING OF PLANAR TRAJECTORIES IN PRESENCE OF OBSTACLES FOR A MANIPULATOR WITH REDUNDANT LINKS

By
ANUPAM BAGCHI

ME

1987

M

BAG

TRA

TM
me/1987/m

B1464



DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
JANUARY, 1987

TRACKING AND PLANNING OF PLANAR TRAJECTORIES IN PRESENCE OF OBSTACLES FOR A MANIPULATOR WITH REDUNDANT LINKS

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By
ANUPAM BAGCHI

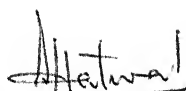
to the
DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
JANUARY, 1987

3 NOV 1987
CENTRAL LIBRARY
Acc. No. 98570

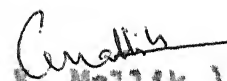
ME-1987-M-BAG-TRA

CERTIFICATE

This is to certify that the present work TRACKING AND PLANNING OF PLANAR TRAJECTORIES IN PRESENCE OF OBSTACLES FOR A MANIPULATOR WITH REDUNDANT LINKS, has been carried out by Mr. S. Anupam Bagchi under our supervision and it has not been submitted elsewhere for a degree.



(H. Hatwal)
Assistant Professor



(A. K. Mallik)
Professor

Department of Mechanical Engineering
Indian Institute of Technology
Kanpur 208 016
INDIA

January, 1987

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude and thanks to Dr. H. Hatwal and Dr. A.K. Mallik, under whose guidance, I did this work. I was given enough independence to try out my ideas and, received encouragement and helpful suggestions throughout my work. I am also indebted to them for giving me full independence to try out my ideas on the robots at the Robotics Centre and use other facilities available there.

Thanks are due to my friends in Hall V and well-wishers who have always lent a helping hand whenever needed, and made my stay during the program an enjoyable experience.

Finally, I wish to thank Swami Anand Chaitanya for careful and efficient typing of this manuscript.

December, 1986.

S. Anupam Bagchi

CONTENTS

<u>Chapter</u>		<u>Page</u>
	LIST OF FIGURES	
	SYNOPSIS	
I.	INTRODUCTION	1
	1.1 Introduction	1
	1.2 Review of Previous Work	2
	1.3 Scope of Present Work	5
II.	FORMULATION OF THE ALGORITHMS	9
	2.1 Introduction	9
	2.2 General Description of Problem and Modelling the Workspace	9
	2.2.1 Problem Statement	10
	2.2.2 Manipulator Configuration to Approximate Given Curve	12
	2.2.3 Preliminary Description of Workspace-Manipulator Modelling	17
	2.2.4 Modelling of Workspace	21
	2.3 Network Description	26
	2.3.1 Establishing Edge Connectivity	26
	2.4 Calculation of Bezier Curve	31
	2.4.1 Choice of Bezier Points	31
	2.4.2 Determining Curve from End- Effector to Present First Edge	39
	2.5 Path Planning of End-Effector in Case of Collision	46

<u>Chapter</u>		<u>Page</u>
III.	FORMULATION AND IMPLEMENTATION OF ALGORITHMS	51
	3.1 Introduction	51
	3.2 Network Representation	51
	3.2.1 Forming Outer Polygon	53
	3.2.2 Forming Triangles	54
	3.2.3 Establishing Edge Connectivity	55
	3.3 Finding Manipulator Configuration	57
	3.3.1 Finding Next End-Effector Position	57
	3.3.2 Finding Manipulator Configuration	58
	3.4 Some Additional Procedures	66
IV.	RESULTS AND DISCUSSIONS	69
	4.1 Introduction	69
	4.2 Object with Three Interior Edges	70
	4.3 Object with Five Interior Edges	85
	4.4 Object Requiring Non-Operational Path Planning	88
V.	CONCLUSIONS	96
	BIBLIOGRAPHY	98
APPENDIX I	BEZIER CURVES	
APPENDIX II	USER'S MANUAL	
APPENDIX III	PROGRAM CODES IN BASIC	

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1.1 Flow-Chart giving an Overview of Algorithm.	8
2.1 Problem Statement	11
2.2 Bezier Curves to Joint Coordinates	11
2.3 Joint Calculation Using Bezier Curves	16
2.4 Defining ray and End-Effector Natural State Number	16
2.5 Defining Present Manipulator State Number and Operational and Non-Operational Path	19
2.6a,b Convex Polygon and Outer Polygon	23
2.7a Outer Polygon Enclosing the Obstacles	23
2.7b List A with Starting Viewing Angle = 0	25
2.8a Outer Polygon with No Obstacle Inside	25
2.8b Outer Polygon with One Obstacle Inside	25
2.8c Outer Polygon with Three Obstacles Inside	25
2.9 Calculation of Head and Tail of Arrow	28
2.10a,b,c Establishing Pointers	28
2.11a Network of 5 Obstacles	32
2.11b Edge Connectivity Graph of Network	32
2.12a Following Pointers from E to O	34
2.12b List Representation with Markers	34
2.13 Adjustment of Final Length	38
2.14 Figure Showing Conditions of a State Change	41
2.15a,b Figure Depicting Case I	43

<u>Figure</u>	<u>Page</u>
2.16 Figure Depicting Case II	44
2.17 Figure Depicting Case III	44
2.18a Figure Depicting Case IV	47
2.18b Solving Case IV	47
2.19 Circumventing Obstacles in Case of Collision	49
3.1 Line-Cuts-Circle	67
3.2 Point-Inside-Triangle	67
4.1 Problem 1: Definition	71
4.2 Modelling of Space	71
4.3 5-Link Manipulator, Positions 3, 6 and 14.	73
4.4 5-Link Manipulator, Positions 18, 29 and 34	73
4.5 5-Link Manipulator, Superimposed Configurations	74
4.6 Bezier Curve and Manipulator Configuration for Case in Problem 1	75
4.7 5 Unequal Links, without Smoothing	77
4.8 5 Unequal Links, with Smoothing	77
4.9a Fifth Joint Angle vs. EE Position Number	78
4.9b Fourth Joint Angle vs. EE Position Number	79
4.9c Third Joint Angle vs. EE Position Number	80
4.10a Last Joint Angle vs. EE Position Number	82
4.10b $(n-2)^{th}$ Joint Angle vs. EE Position Number	83
4.11 6-Link Manipulator, Equal Link Lengths	84
4.12 5-Link Manipulator, Equal, Base Nearer, Positions 6, 13 and 27	84
4.13 Problem 2: Definition	86
4.14 Modelling of Workspace	86

<u>Figure</u>		<u>Page</u>
4.15	5-Link Manipulator, Positions 2, 8 and 13	87
4.16	5-Link Manipulator, Positions 19, 24 and 31	87
4.17a,b,c	Demonstrating Markers	89
4.18	Physical Description of Task: Problem 3	90
4.19	Problem Definition in 2-D	90
4.20	Modelling of Workspace	90
4.21	Five Link Manipulator, Positions 6, 12 and 18	92
4.22	Operational and Non-Operational Path	92
4.23	5-Link Manipulator, Positions 20, 25 and 27	93
4.24	Superimposed Configurations	93
4.25	5-Link Manipulator, Positions 35, 39 and 45	94

SYNOPSIS

The present work attempts inverse solutions for manipulators with redundant links. Such problems have been traditionally solved using either some optimization techniques or pseudo-inverses of Jacobians. This work presents a new approach by modelling the workspace with obstacles with respect to the manipulator. The workspace has been represented as a network. An algorithm is presented which finds a safe configuration of the manipulator by following a collision free path through the network. This path is first represented as a smooth curve from end-effector to the base. Depending on the links, the manipulator joints are then located as close as possible to the curve. Three different problems are discussed illustrating the salient points of this algorithm.

CHAPTER I

INTRODUCTION

1.1 INTRODUCTION:

Robots are being used to handle increasingly complex tasks day by day. Most of the regular jobs need upto five or six degrees of freedom manipulators. However, many situations demand a manipulator with more manoeuvre ability. For example, in a spray painting robot, where the cavities of a car body are to be accessed, it is necessary that the manipulator should have extra degrees of freedom (or redundancies). Stated in simpler terms, if the manipulator arm has to circumvent some obstacles in the workspace, extra degrees of freedom are to be provided.

There are industrial applications today where the manipulators have more than six degrees of freedom. For example, the elephant trunk type manipulators are used for spray-painting automobiles. Presently, the robots are programmed with lead-through teaching programming, where the manipulator is guided through various stages and at each stage the joint positions are stored.

It is naturally desirable that given the obstacles in the workspace, the manipulator configuration should be obtainable through some inverse solutions. This may become

essential if the manipulator is being used in a hazardous environment. It is easy to see that the inverse solutions for such manipulators is extremely complicated because, out of infinite solutions (arising due to redundancies), only those solutions are valid which avoid the collision between the obstacles and the manipulator. Thus the inverse solution necessitates the modelling of workspace (i.e. description of obstacles in the workspace) together with the modelling of manipulator. As stated earlier, there will be more than one manipulator configuration for each end-effector position. Thus the inverse solution procedure can be roughly stated to have two parts: given the end-effector position, first find the safe configurations and then choose one such safe solution. Most of the algorithms tackle both the parts simultaneously.

1.2 REVIEW OF PREVIOUS WORK:

The approach to this problem by previous authors, has centered around specifying a distance function that varies inversely with the proximity of any link with an obstacle in space. Loeff and Soni (1975) [1] proposed an algorithm that gave an incremental configuration of the manipulator for each incremental end-effector position. The "influence" of each obstacle on the links was measured in terms of the relative distance between them. Khatib, 1985 [2] based his solution on the "artificial potential field" concept. In this approach,

collision avoidance, traditionally considered a high level planning problem, was effectively distributed between different levels of control. The philosophy of the artificial potential field approach can be schematically described as follows. The manipulator moves in a field of forces. The position to be reached is an attractive pole for the end-effector, and obstacles are repulsive surfaces for the manipulator parts. The manipulator obstacle avoidance problem was formulated in terms of collision avoidance of links, rather than joints. Link collision avoidance was achieved by continuously controlling the link's closest point to the obstacles. Kircanski and Vukobratovic (1984) [3] developed an algorithm based on performance indices in the space of joint coordinates and external coordinates, which prevent any of the manipulator links from entering a forbidden zone around the obstacle. The performance index was based on penalizing motion of those joints which influence the motion of the closest point of a link to an obstacle. Maciejewski and Klein, (1985) [4] considered the obstacle avoidance in the context of a specified end-effector trajectory with a specified velocity. The approach was to determine the required joint angle rates for the manipulator under the constraints of multiple goals. The primary goal was described by the specified end-effector trajectory and secondary goals described the obstacle avoidance criteria. The primary goal and the secondary goal were described by the equations

$$J_e \dot{\theta} = \dot{x}_e$$

and $J_o \dot{\theta} = \dot{x}_o$

where,

J_e = end-effector Jacobian

J_o = obstacle avoidance point Jacobian

\dot{x}_e = specified end-effector velocity, and

\dot{x}_o = specified obstacle avoidance point velocity

The solution of the desired end-effector trajectory composes the particular portion of the solution. In the redundant case this guarantees the exact desired end-effector velocity with minimum joint velocity norm. The homogeneous solution sacrifices the minimum norm solution to satisfy a different goal, that of obstacle avoidance.

Approaches based on space representations have been dealt by Brooks and Lozano-Perez (1985) [5] and Brooks (1983)[6] but they have essentially dealt with finding a continuous collision free path for an object in a space cluttered with obstacles. The obstacles were represented as convex polygonal bodies or as a union of polygonal shapes. Space was either represented as a union of generalized cones [6] or by a recursive cellular representation of configuration space [5]. Giralt [7] reported a space representation scheme used for a mobile navigational robot. Space was represented at two levels: 1) the topological level where places are nodes, and connectors are arcs in a connectivity graph, and

2) the geometric level which assigns dimensions to the elements of the connectivity graph: width to the connectors and boundary dimensions to the places.

In the author's knowledge, use of the space representation scheme to solve for the robot configuration in the presence of obstacles, has not been earlier attempted.

1.3 SCOPE OF PRESENT WORK:

In general the end-effector needs only two degrees of freedom for plane motion (without considering orientation of last link) in a workspace free of obstacles. If obstacles are to be avoided, some extra links are needed for obstacle avoidance manoeuvre. This work does not deal with determination of optimum redundancies required for a particular task. It is assumed that few extra links are provided in the manipulator. This work concentrates on finding safe manipulator configuration for a given end-effector position. In some cases, it may be necessary to retract the manipulator, circumvent an obstacle and reach the desired end-effector position. Thus a path planning for end-effector is needed. This work also proposes algorithms for such path planning.

Some of the disadvantages encountered in earlier schemes are presented below.

1) Use of distance functions inherently leads to optimization which has to be done for each end-effector position. In

some cases this may lead to oscillations around the optimum configuration and appropriate blending of homogeneous solutions may be required [4]. Moreover, convergence to a local minima is likely and has to be carefully avoided.

ii) When the number of redundancies are large, the computation time is likely to increase, because the formulation is dependent on the number of links.

iii) While tracking an end-effector trajectory, a collision may become imminent. In such cases, a new configuration may be required to access the same point. Finding a new configuration and calculating the additional end-effector path required to change the present configuration has been entirely neglected so far.

The present work proposes a formulation which is independent of the number of links. This scheme would thus be advantageous when redundancies are large. Distance functions have not been used and emphasis has been laid on a good topological representation of workspace having obstacles, in the form of a directed graph. A search through this graph yields a rough path which is then refined to a smooth curve. This curve is used to find out the manipulator configuration. A scheme has been developed to represent the status of the manipulator with respect to the obstacles around it. This scheme is found to be very useful when a given end-effector position seems inaccessible. Then the task is threefold:

a new safe configuration of the manipulator is to be determined, a trajectory for the end-effector is to be planned such that the manipulator may retract from the old configuration and reach the new one, and finally at each incremental position of end-effector the manipulator is to be in a safe configuration.

The summary of the approach of this work is presented in flow-chart of Fig. 1.1.

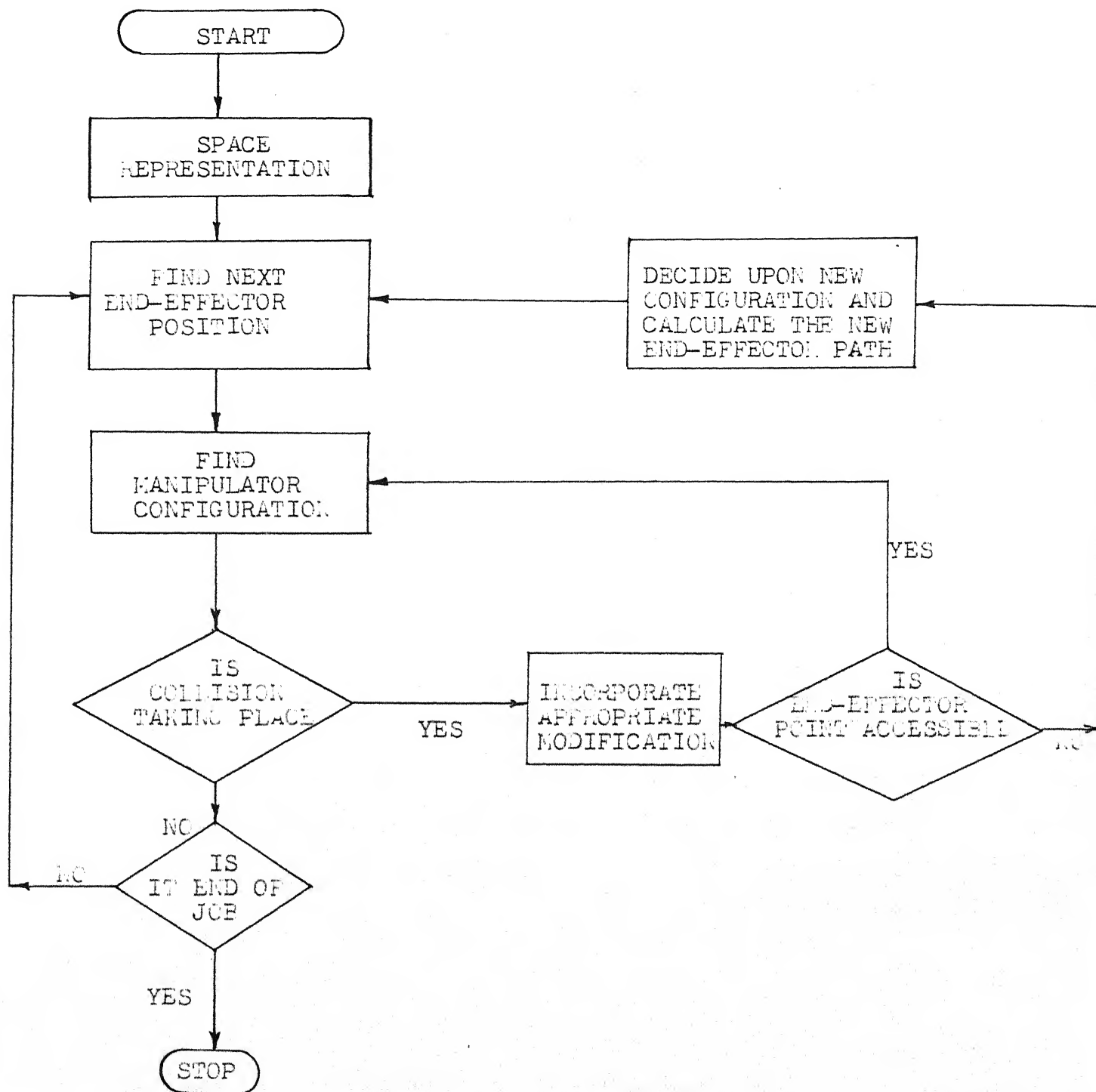


Fig. 1.1: Flow-Chart giving an Overview of Algorithm.

CHAPTER II

FORMULATION OF THE ALGORITHMS

2.1 INTRODUCTION:

This chapter gives detailed descriptions of steps in the formulation of algorithms. As stated earlier in Sec. 1.3 the manipulator has been modelled as a smooth curve. Sec. 2.2 gives a general description of the problem and the strategy used to model the workspace. This section also describes how the smooth curve is used to calculate the manipulator configuration. The network representation of the workspace in the form of a graph is taken up in Sec. 2.3. Using the above network, a smooth curve which avoids all obstacles is calculated. This has been detailed in Sec. 2.4. Finally, in Sec. 2.5, path planning of the end-effector in case of collision has been considered.

2.2 GENERAL DESCRIPTION OF PROBLEM AND MODELLING THE WORKSPACE:

Before formal presentation of the complete method, we describe in this section, through some examples, the objectives, the approach to formulate the algorithms and some terminology used in this work.

2.2.1 Problem Statement:

The point O represents the base of a manipulator and the end-effector is indicated by E in Fig. 2.1. There are obstacles shown by hatched areas in the workspace. An ideal flexible single link manipulator could have taken the shape shown by the dashed line. With rigid links, obviously, more the links, better would be the shape or configuration taken by the manipulator. With five links, one possible configuration could be as shown in Fig. 2.1 by firm lines.

In the present work the manipulator and workspace is assumed to be planar. All the obstacles are assumed to be circular. First step in the solution is to find a (there could be many) curve from O to E avoiding all the obstacles and the second step is to fit the links to approximate this curve. Determining a curve from O to E is the major part of the work and is dealt with in all following sections in steps. Here, we describe a procedure to obtain the link positions and orientations which approximately fit a given curve of length equal to the sum of lengths of all links.

The following convention is followed to label the links and the joints. The first link towards the base O will be referred to as the "first" link, and that towards the end-effector will be called the "last" link. The last link number

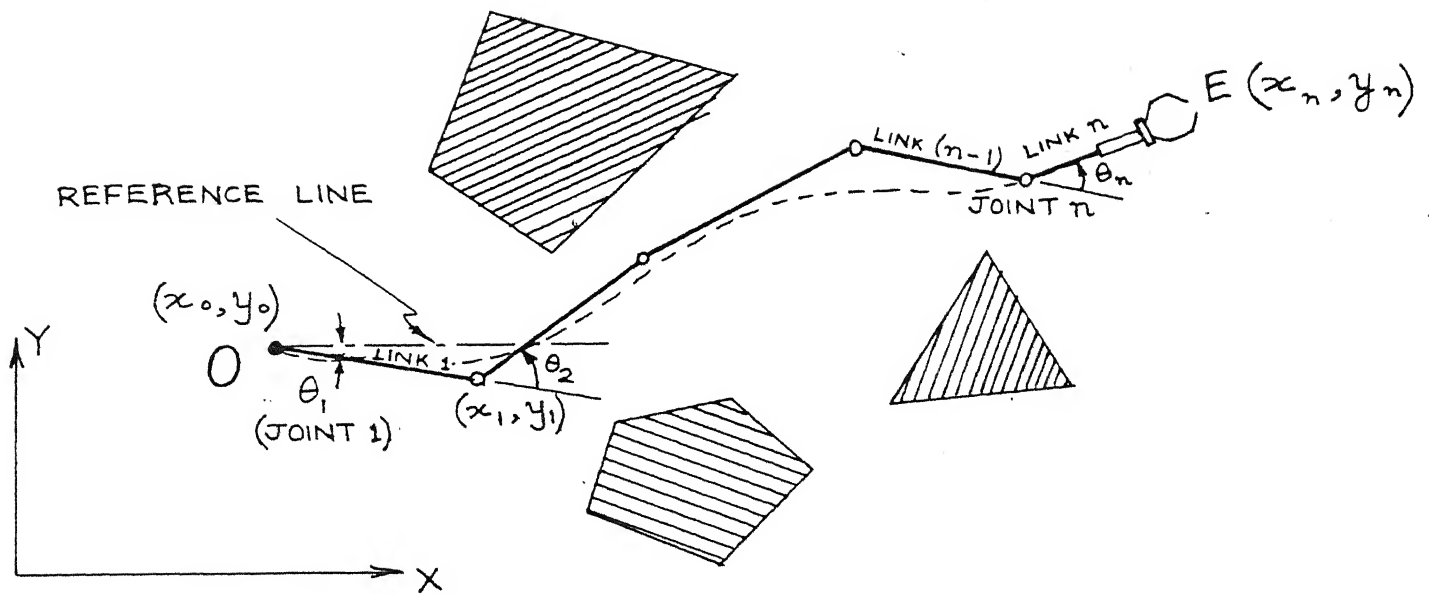


FIG 2.1 : PROBLEM STATEMENT

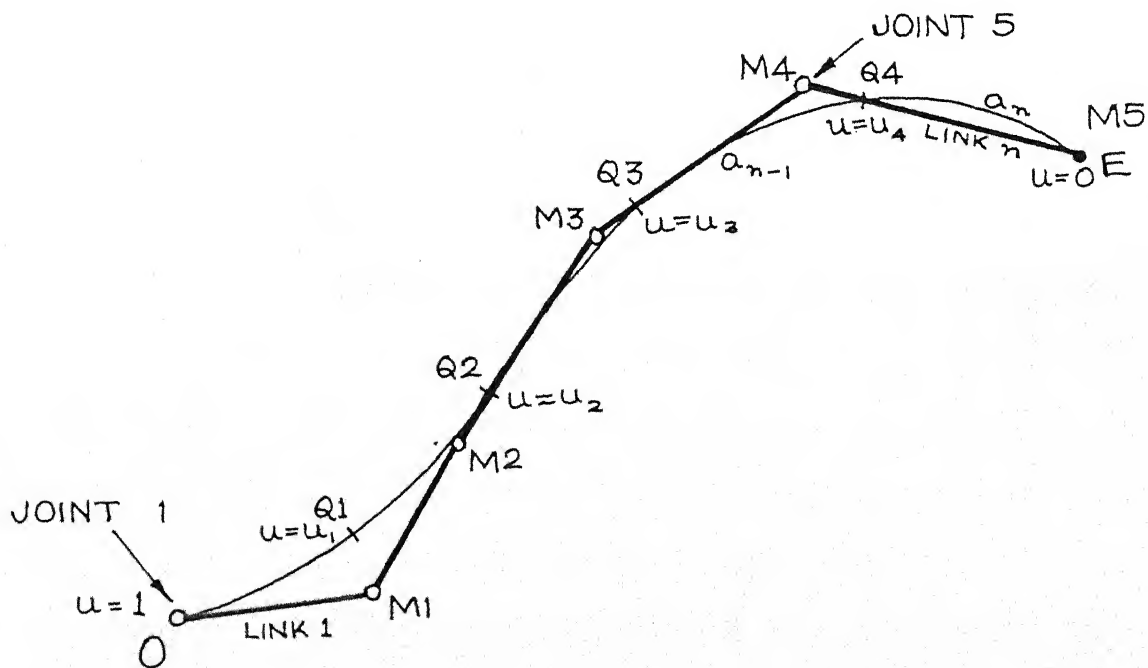


FIG 2.2 : BÉZIER CURVES TO JOINT COORDINATES

will be n , where n = total number of links. The link lengths will be represented by a_i which is the length of the i^{th} link. The joint between link i and link $i-1$ will be called the i^{th} joint. Thus the n^{th} joint is the joint which connects link $(n-1)$ and link n . Joint number 1 is the joint at the base O .

The joint coordinate implies the relative angle between the two links connected by this joint. So, joint coordinate θ_i is the angle between link $(i-1)$ and link i . Joint coordinate 1 is measured from some reference line fixed at the base O . Besides the joint coordinates, it is also necessary to specify the coordinates of the joints, which is the x - y coordinate of the joint location in the plane of movement, with respect to any coordinate system attached to it. Joint 1 will have the coordinates (x_{1-1}, y_{1-1}) . The coordinates of the end-effector position E will be (x_n, y_n) , and those of the base O will be (x_0, y_0) .

2.2.2 Manipulator configuration to Approximate a Given Curve:

In this section we assume that a curve from the end-effector position E to the base O has been given to us. The curve is a Bezier Curve which has been generated by suitably choosing some Bezier Points in the workspace. (For description of Bezier Curve, see Appendix I). The details of choosing the Bezier Points are elaborated later in Section 2.4.

The resulting curve is of a definite length which is the sum of all the link lengths. Thus

$$L = \sum_{i=1}^n a_i \quad (2.1)$$

where a_i represents the length of the i^{th} link. The base of the robot O , is assumed to be invariant with respect to time. Referring to Fig. 2.2, consider a smooth curve from O to E . A parameter u varies along the curve and is equal to zero at E and 1 at the base O . Our objective is to determine the coordinates of the joints (i.e., x-y coordinates of the joint location in the plane). With infinite links, the joints would have been located exactly on the Bezier curve. The actual joints ought to lie as close as possible to the ideal point which lies on the smooth curve. The "ideal" point is calculated as follows.

Suppose we want to find out the location of the i^{th} joint. Then the procedure would be to find a suitable point Q_i on the Bezier curve, and then depending on the location of Q_i find another point M_i which is close to Q_i . Thus Q_i would represent the "ideal" point and M_i the actual coordinates of the joint location of the $(i+1)^{\text{th}}$ joint. We first find M_n then M_{n-1} , M_{n-2} and so on upto M_1 . So, the joint-locations are found from the end-effector side, finally ending at the base O . M_n would be the end-effector position E while M_0 would be the coordinates of the base O . We assume that we have only one Bezier Curve between O and E .

The ideal point Q_1 is calculated as follows. First we have to find out the value of the parameter v which corresponds to point Q_1 . A parameter v_1 is first calculated as,

$$v_1 = \frac{L - \sum_{j=1}^1 a_j}{L} \quad (2.2)$$

The parameter u_1 is now obtained from the formula

$$u_1 = \gamma \sin \pi v_1 \quad (2.2 b)$$

where γ represents the maximum measure of the correction factor necessary to calculate u_1 from v_1 . The spacing of the Bezier Points is utilised to obtain γ . Let the order of a certain Bezier curve be Z . Then there are $Z + 1$ Bezier points. Let the shortest distance between the $(k-1)^{th}$ and the k^{th} Bezier Point be l_k . Then we associate a parameter w_k with each Bezier Point such that

$$w_k = \frac{\sum_{j=1}^k l_j}{\sum_{j=1}^Z l_j} \quad (2.2c)$$

Now γ is calculated as,

$$\gamma = \rho \times \left\{ \max \left[\frac{k}{Z} - w_k \right] \right\}, \quad k = 1, Z-1 \quad (2.2d)$$

where ρ is a scaling factor. A value of $\rho = 0.5$ has been found to give good results in the implementation.

Now, given u_1 , the ideal point Q_1 is calculated on the curve by using,

$$\vec{r}_1 = \sum_{k=0}^z \vec{r}_k z_{C_k} (1 - u_1)^{z-k} v_1^k \quad (2.3)$$

where, z represents the order of the Bezier Curve, \vec{r}_k is the vector representation of the k^{th} Bezier Point and \vec{r}_1 is the vector which defines point Q_1 . This is the "ideal" point taken for the i^{th} joint. The actual point has to lie as close as possible to Q_1 . One possible scheme is to impose the condition that the point M_1 has to lie on the straight lines joining M_{i+1} to Q_1 , since we are approaching from the end-effector side. The procedure will be adopted only for $i > 2$ since the position taken by joint 2 can be calculated directly. So, for $2 < i < (n-1)$, the coordinates of the i^{th} joint are

$$x_{M_i} = x_{M_{i+1}} + \text{sgn}(x_{Q_1} - x_{M_{i+1}}) \frac{a_{i+1}}{\sqrt{1+m^2}} \quad (2.4a)$$

$$y_{M_i} = y_{M_{i+1}} + m(x_{M_i} - x_{M_{i+1}}) \quad (2.4b)$$

where m is the slope of the line $Q_1 M_{i+1}$

$$m = \frac{(y_{Q_1} - y_{M_{i+1}})}{(x_{Q_1} - x_{M_{i+1}})} \quad (2.4c)$$

It may so happen that the joint limitation of the $(i+1)^{\text{th}}$ joint is reached when M_i is being calculated (See Fig. 2.3). In that case the link $(i+1)$ is put to the limiting position

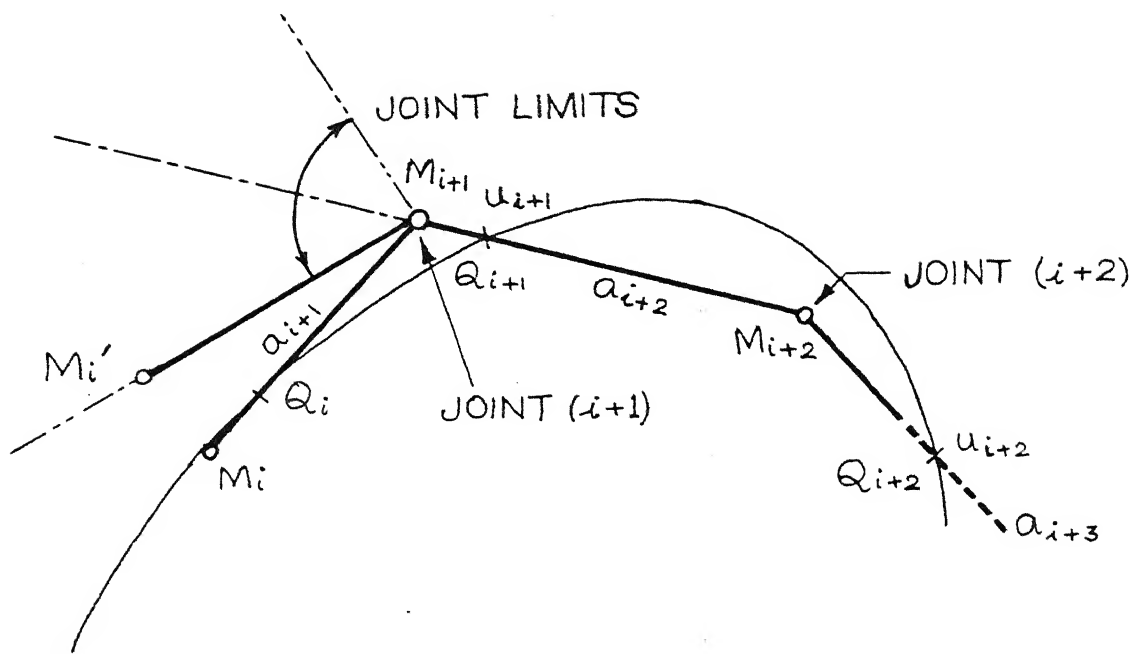


FIG 2.3 : JOINT CALCULATION USING BÉZIER CURVES

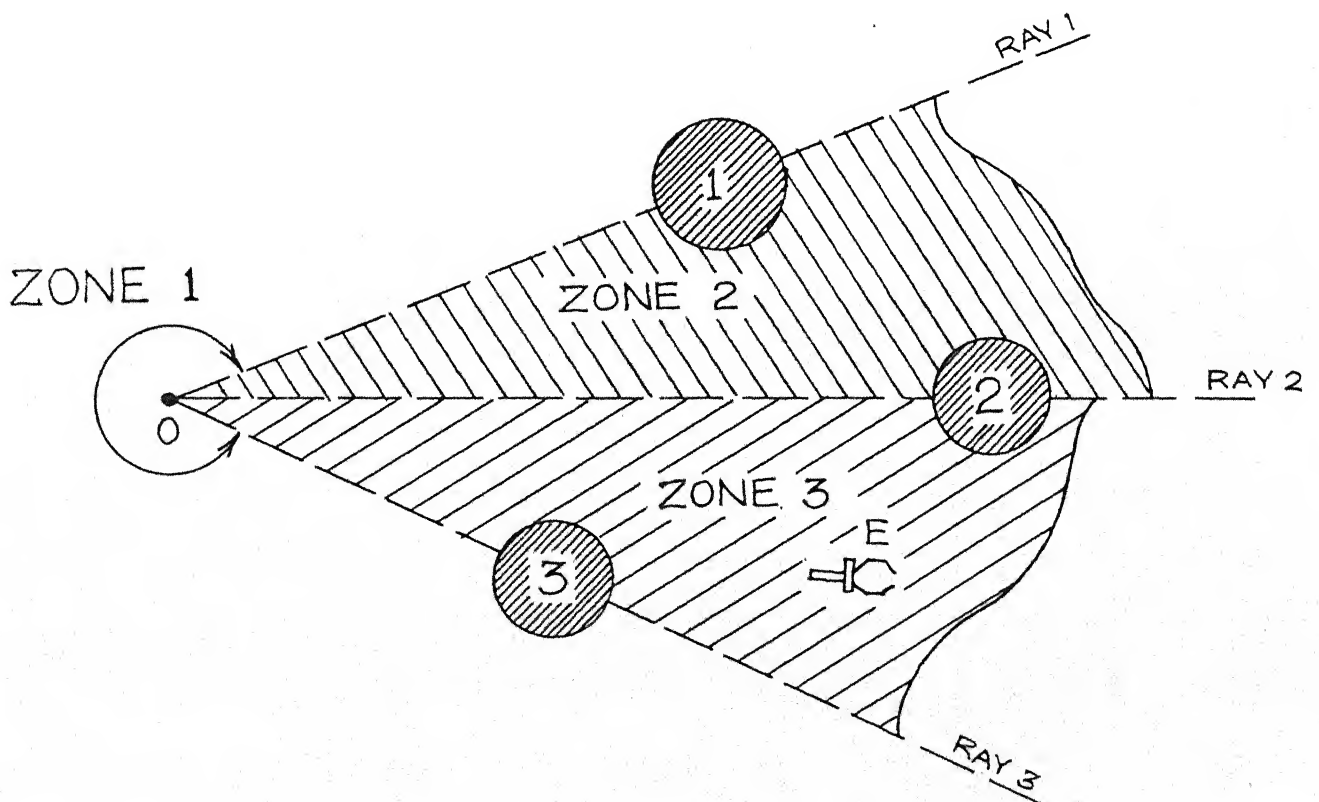


FIG 2.4 : DEFINING RAY AND END-EFFECTOR NATURAL STATE NUMBER

and a new position M_1' is calculated. As stated earlier, the location of joint 2 (i.e. M_1) can be found directly by calculation and it can take only two positions. The one which is nearer to Q_1 is taken as the position M_1 .

2.2.3 Preliminary Description of Workspace-Manipulator Modelling:

Before we present the modelling of workspace and the manipulator description with respect to it, an example is taken up to briefly illustrate the idea behind the modelling. In the process, the terms used in the present work are also defined.

Fig. 2.4 shows three obstacles all of which are circular. The base of the manipulator is at O . The centre of each obstacle is joined to the base O and the line $O(i)$ will be called a ray. Thus the i^{th} ray is the ray found by joining O to the centre of obstacle i . The obstacles are numbered depending on the angle of its ray. The obstacles are ordered in clockwise sense. Going clockwise from the i^{th} ray to the $(i+1)^{\text{th}}$ ray, the zone between these is termed the $(i+1)^{\text{th}}$ zone. The entire zone from ray 3 to ray 1, going clockwise, will be the zone 1. If the end-effector E lies in the i^{th} zone, then we say that the "end-effector natural state" is i . In Fig. 2.4, the "end-effector natural state" is 3 since it lies in zone 3.

An edge " ij " ($i \neq j$) is the line joining the i^{th} and j^{th} obstacles. The edges are always numbered so that $i < j$. Thus if the workspace has n obstacles, then nC_2 edges are possible.

It is shown later that all the edges are not needed to model the workspace. Fig. 2.5 shows a workspace with five obstacles. The workspace is modelled by selecting edges which form non-overlapping triangles such that all the obstacles are encompassed. Sec. 2.2.4 describes the complete procedure to obtain a practical set of non-overlapping triangles. At present let us consider the triangles shown in Fig. 2.5. Each triangle is identified by an ordered set ijk where i , j and k represent the obstacle numbers and $i < j < k$. Thus the triangles 123, 234 and 345 form a set of non-overlapping triangles encompassing all obstacles. The edges $i(i+1)$ are called "outside edges". Thus edges 12, 23, 34 and 45 represent the outside edges in Fig. 2.5.

The "natural edge" is that "outside edge" which corresponds to the "end-effector natural state". Thus if the "end-effector natural state" is i , then the "natural edge" would be $(i-1)(i)$.

We also need a description of the manipulator. This is done by assigning a "present manipulator state number" to it. The present manipulator state number is defined by taking up two cases below. In case I the manipulator does not intersect any outside edge as shown by position E_5 in Fig. 2.5. Then the present manipulator state number is the end-effector natural state number. Thus for position E_5 in Fig. 2.5, the present manipulator state number is 5 since it lies in zone 5. In the second case, the manipulator intersects one or more

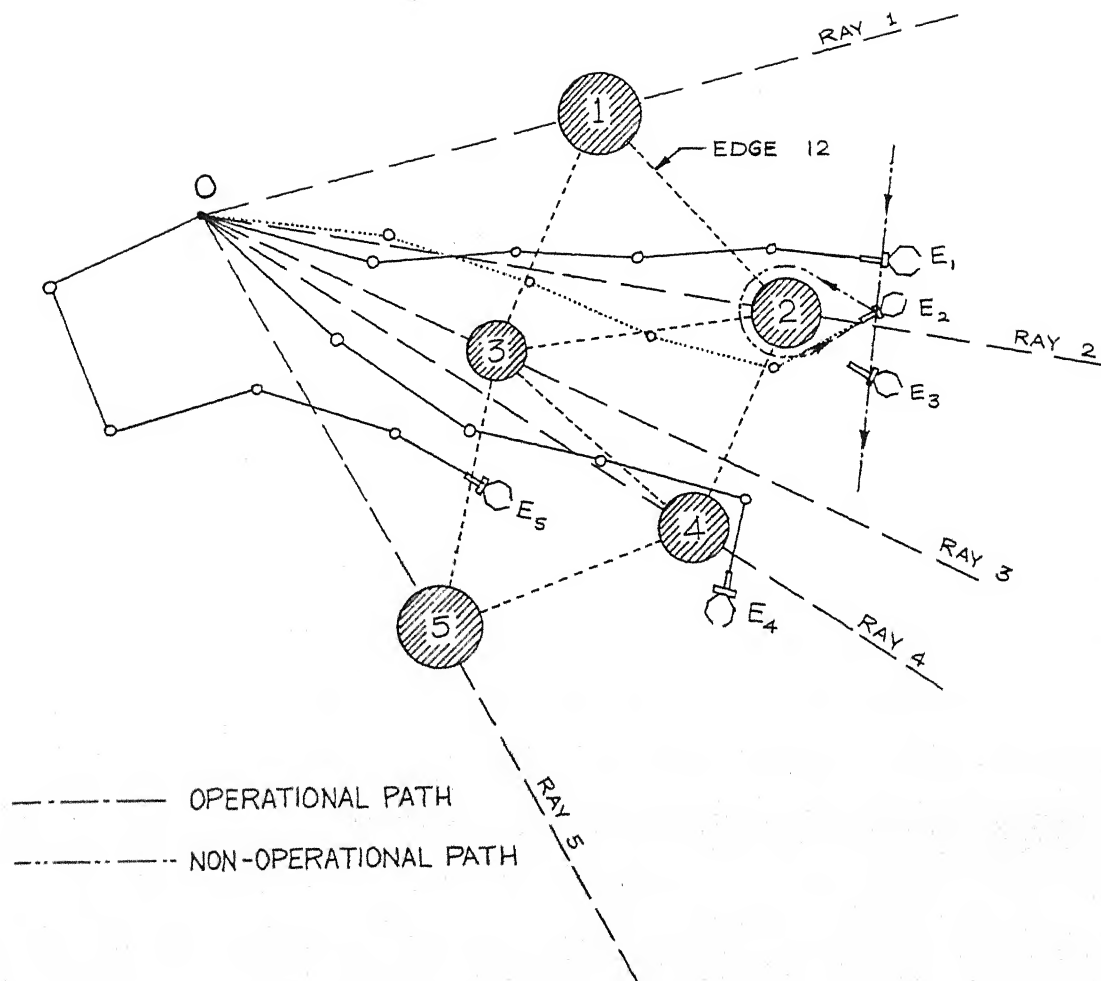


FIG 2.5 : DEFINING PRESENT MANIPULATOR
STATE NUMBER AND OPERATIONAL AND
NON-OPERATIONAL PATH

outside edges (as in position E_4 in Fig. 2.5). Then we note the outside edge which is intersected first while going from base to end-effector. If the edge $i(i+1)$ is this edge, then the present manipulator state number is $(i+1)$. Thus in Fig. 2.5 the present manipulator state number for position E_4 is 4, since the "outside edge" 34 is being intersected first, if we traverse along the manipulator from O to E. The outside edge which determines the present manipulator state number is called the "present first edge" and is used later to generate the first iterative configuration of the manipulator.

We will now discuss something about path planning and introduce two new terms, "operational path" and "non-operational path". The operational path will be described by the user during which the actual operation to be performed by the robot will be done. The non-operational path will lead to idle time, but, which in some cases will be essential to avoid a collision or to reach a remote point in the path. Referring again to Fig. 2.5 let us say that an operation has to be done by tracking points on the path shown by $E_1E_2E_3$. Thus E_1 to E_2 and E_2 to E_3 form the "operational path".

The manipulator configuration at E_1 is shown in Fig. 2.5. For this position, the present manipulator state is 2 and the end-effector natural state is also 2. The end-effector now moves to position E_2 maintaining the present manipulator state. The configuration at E_2 shown with the dotted line has the present manipulator state number equal to 3. When the end-

effector moves from position E_2 to E_3 in present manipulator state number 2, it fouls with obstacle 2. The obvious solution then is to move from E_2 , approach obstacle 2, retract back and after circumventing obstacle 2 again reach position 2. It would thus take the configuration shown by the dotted line in Fig. 2.5 and would have the present manipulator state number equal to 3. From E_2 , the manipulator can now access point E_3 in present manipulator state number 3. The complete path then is E_1 to E_2 , E_2 to E_2 and E_2 to E_3 . The path from E_2 to E_2 circumventing obstacle 2 is called the "non-operational path" and this act is called a "state change operation".

2.2.4 Modelling of Workspace:

The first step in the solution procedure requires a suitable path for the Bezier Curve of a given length from the base 0 to the end-effector E. Finding Bezier points in the workspace should not require time consuming computations because this search has to be done repeatedly for each position of the end-effector. A systematic representation of the workspace with obstacles is described in the following paragraphs, through which the Bezier Points are determined quite efficiently.

The obstacles which are present in the workspace yield few regions which are free and which would be possible passages for the Bezier Curve. It is our aim here to represent the space around the obstacles in such a way that such regions could be easily identified and followed till the space around 0 is

reached. For this, it is necessary to enclose the working space and divide it into smaller non-overlapping segments. Later in this chapter, we will set a procedure to establish "pointers" between adjacent segments. A feasible path could be found by starting from the segment nearest to the end-effector following consecutive pointers.

The first task is to form a polygon enclosing all obstacles. The polygon is not strictly the convex polygon of the obstacles. Fig. 2.6a shows a workspace with 5 obstacles. The convex polygon is 1452. The edge 14 grazes through obstacle 3 since 3 is just inside the polygon. Such cases are to be avoided as they might cause problem while setting pointers. It is desirable to include obstacle 3 in the boundary of the polygon. Thus, our requirement is satisfied if the boundary is 13452. This is represented in Fig. 2.6b. The resulting polygon is no more convex and we call it the "outer polygon" to distinguish it from a strictly convex polygon.

The procedure adopted for forming an outer polygon is explained below. First, a large circle which encloses all the obstacles is formed. This circle has its centre at C (Fig. 2.7a) where,

$$x_c = \frac{(x_{\min} + x_{\max})}{2} \quad (2.5a)$$

$$y_c = \frac{(y_{\min} + y_{\max})}{2} \quad (2.5b)$$

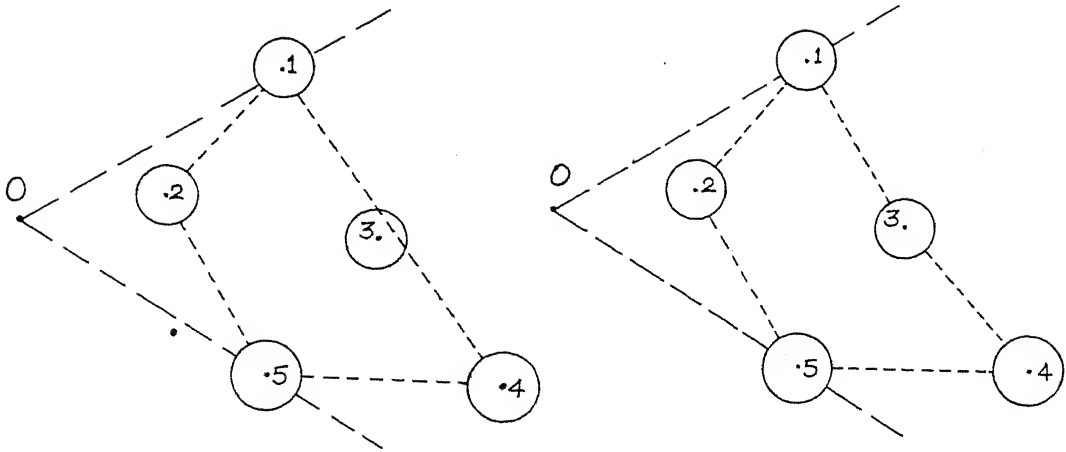


FIG 2.6a: CONVEX POLYGON FIG 2.6b: OUTER POLYGON

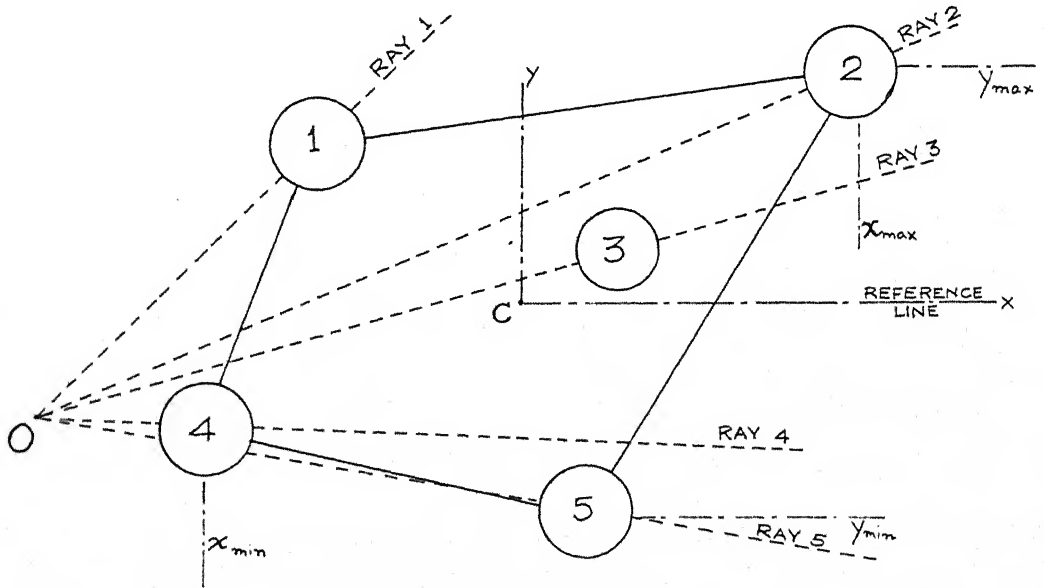


FIG 2.7a: OUTER POLYGON ENCLOSING THE OBSTACLES

A point on the circumference of this circle is now chosen to act as a viewing point for all obstacles. The angle measured from some reference line (in 0° to 360° range) at which the first viewing point is chosen will be called the "starting viewing angle". The obstacles are viewed from the present viewing point and the nearest obstacle is stored in a list, say A. The viewing point is changed by moving along the circumference of the circle in any one direction. The viewing angle is incremented at each step by,

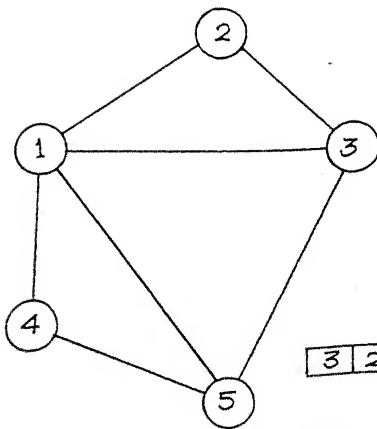
$$\Delta\theta = \frac{360^{\circ}}{n} \quad (2.6)$$

where n = number of obstacles. The nearest obstacle is again found and appended to the list if the last entry was different. Fig. 2.7b shows the list A obtained from the situation shown in Fig. 2.7a, with the starting viewing angle equal to 0° .

Having obtained the outer polygon, the next step is to segment this into smaller parts. We have chosen to break it up into non-overlapping triangles. The procedure to do this is not unique and any suitable method may be followed without affecting the results much. Herein three cases are identified which are depicted in Fig. 2.8a, b, c. Fig. 2.8a shows the case where there are no obstacles inside the polygon. In this case, sets of three points are chosen from the list A in sequence, and are then grouped together to form triangles. Obviously, the triangles would depend entirely on the list A, and in case the triangles prove to be unsatisfactory, they

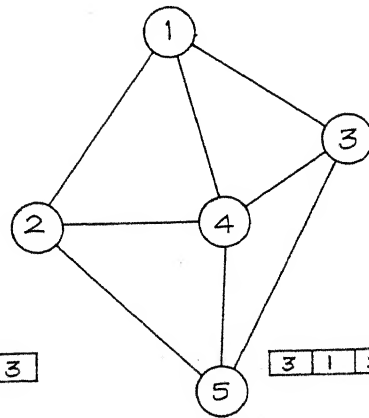
2	1	4	5	2
---	---	---	---	---

FIG 2.7 b : LIST 'A' WITH STARTING
VIEWING ANGLE = 0



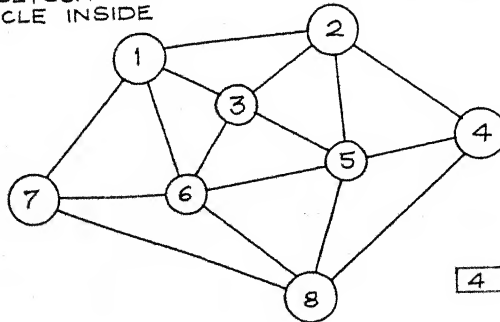
3	2	1	4	5	3
---	---	---	---	---	---

FIG 2.8a : OUTER POLYGON WITH
NO OBSTACLE INSIDE



3	1	2	5	3
---	---	---	---	---

FIG 2.8b : OUTER POLYGON WITH
ONE OBSTACLE INSIDE



4	2	1	7	8	4
---	---	---	---	---	---

FIG 2.8c: OUTER POLYGON WITH THREE OBSTACLES INSIDE

may easily be changed by changing the starting viewing angle. A triangle which is too flat or too much scalene has to be avoided because it creates ambiguous situations while setting pointers.

In the second case, as shown in Fig. 2.8b there is one obstacle inside the polygon. This gives rise to as many triangles as there are unique elements in the list and each is obtained by joining pairs of adjacent elements in the list to the left-over element.

When the number of obstacles, which are not connected by the outer polygon edges, become more than one, the solutions (i.e. the formation of non-overlapping triangles) tend to be more varied and any suitable procedure can be adopted. One method is to identify all elements (i.e. objects) not in the list and form a polygon out of them. The vertices of the inner polygon are joined to the nearest vertices of the outer polygon such that the edges do not intersect. The above procedure can be used recursively till all the points are exhausted. Fig. 2.8c illustrates a simple cases where the left out obstacles are three. With two obstacles inside the outer polygon, the inner polygon reduces to a line.

2.3 NETWORK DESCRIPTION:

2.3.1 Establishing Edge Connectivity:

The next step in the solution procedure is to establish rough directions for a path from E to O. This is not to be confused with the path actually traced by the end-effector.

While trying to find a path from E to O, we are merely looking for positions and orientations which the manipulator links could take. The orientation of a link is the angle it forms with respect to a fixed reference line. The directions for a path from E to O are set as follows. At the middle of each edge and perpendicular to it, an arrow is established. It is desired that the orientation of the link closest to that edge should be close to the direction set by the arrow. The task of searching a path would be expedited if the connectivity of the edges are established through pointers. The network can thus be represented as a graph whose nodes are the edges forming the triangles.

Before we go into a discussion of the edge connectivity graph, we describe the mode of calculation of the arrow coordinates. Referring to Fig. 2.9, consider an edge with two obstacles at the end points A (x_1, y_1) and B (x_2, y_2) with radii p_1 and p_2 , respectively. The objective is to calculate the coordinates of the head H (x_H, y_H) and tail T (x_T, y_T) of the arrow. The coordinates of the mid-point R are obtained from

$$x_R = \frac{x_{s_1} + x_{s_2}}{2} \quad (2.7a)$$

$$y_R = \frac{y_{s_1} + y_{s_2}}{2} \quad (2.7b)$$

where s_1 and s_2 are the points at which the circles at A and B

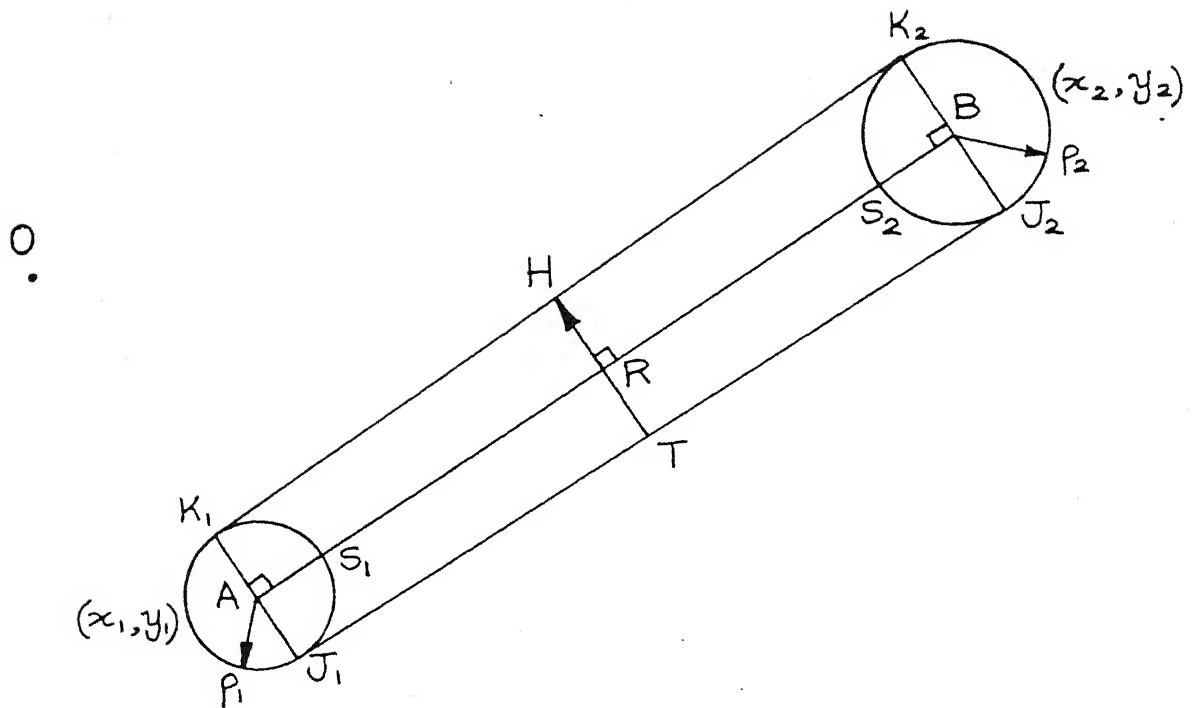


FIG 2.9: CALCULATION OF HEAD
AND TAIL OF ARROW

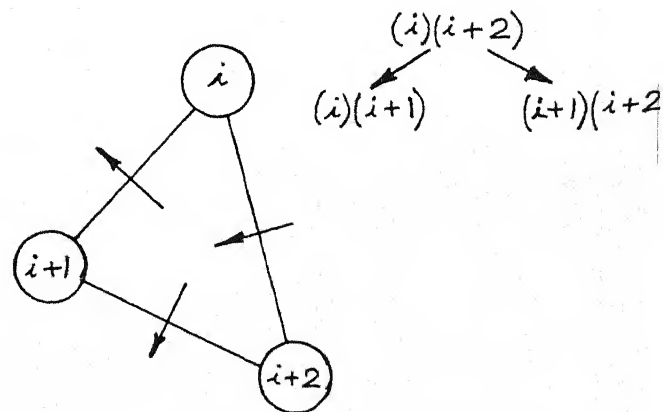
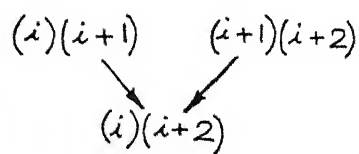
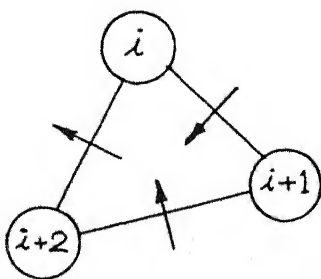


FIG 2.10a: ONE POINTING OUTSIDE FIG 2.10b: TWO POINTING OUTSIDE

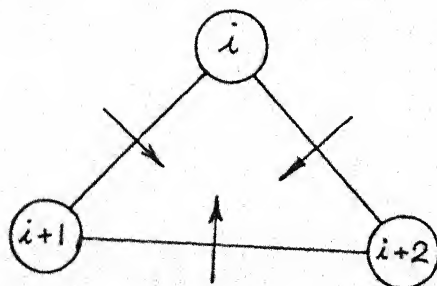


FIG 2.10c: NONE POINTING OUTSIDE

cut the line AB. HT is perpendicular to line S_1S_2 . H is obtained by the point of intersection of line K_1K_2 with the perpendicular at R while T is obtained by its intersecting point with line J_1J_2 . Both K_1J_1 and K_2J_2 are perpendiculars at A and B to line AB.

Thus if,

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad (2.8)$$

then,

$$x_{K_1} = x_1 + \sqrt{\frac{m^2 r_1^2}{1 + m^2}} \quad (2.9a)$$

$$y_{K_1} = y_1 - \frac{1}{m} (x_{K_1} - x_1) \quad (2.9b)$$

and,

$$x_{K_2} = x_2 + \sqrt{\frac{m^2 r_2^2}{1 + m^2}} \quad (2.10a)$$

$$y_{K_2} = y_2 - \frac{1}{m} (x_{K_2} - x_2) \quad (2.10b)$$

With these,

$$x_H = \frac{y_{K_1} - y_R - \frac{x_R}{m} - \epsilon x_{K_1}}{(-\frac{1}{m} - \epsilon)} \quad (2.11a)$$

$$y_H = y_R - \frac{1}{m} (x_H - x_R) \quad (2.11b)$$

where,

$$\epsilon = \frac{(y_{K_2} - y_{K_1})}{(x_{K_2} - x_{K_1})} \quad (2.12)$$

Similarly T can be calculated. It turns out that this procedure is simpler than calculating the common tangents to circles at A and B and then finding the point of intersection with the perpendicular at R .

The choice of the direction of the arrow needs careful scrutiny. If the obstacles in the workspace are such that these can be idealized by physically unconnected circular objects, then the head of the arrow is always nearer to the base O than the tail. This rule can thus help in deciding the direction of the arrow in this case. If some of the obstacles are connected to each other with a wall which the manipulator cannot cross, then that edge will be called a "barrier edge." The basic philosophy underlying the choice of the head and tail of the arrow in such cases, is that on following the head from the tail side, the manipulator should not collide with a barrier edge. In the present implementation, barrier edges have not been considered and if such a case arises, the data representing the network has to be modified manually.

The next job is to establish the connectivity of the edges. To do this, one triangle at a time is taken, which has all the arrow directions assigned. Three cases may arise here as shown in Fig. 2.10a, b and c. Fig. 2.10a shows a triangle in which only one pointer points outside the triangle while the other two point inside. The vertices have been numbered as 1, 1+1 and 1+2. In this case the pointers are $(1)(1+1) - (1)(1+2)$ and $(1+1)(1+2) - (1)(1+2)$. The symbol $-$ is read as "points to".

Fig. 2.10b shows the case 2 where two pointers are pointing outward. Thus, the representation is $(1)(1+2) - (1)(1+1)$ and $(1)(1+2) - (1+1)(1+2)$. Case 3, as shown in Fig. 2.10c, is a case when the base of the manipulator is inside a triangle and no pointers would be established here. There would be of course, other pointers belonging to these network that point to these edges.

Finally, an example illustrates the representation of a network. Fig. 2.11a shows a region with 5 obstacles. The triangles that have been formed are 123, 234, 345 and 135. The edges that have been identified are 12, 24, 45, 23, 34, 13, 35 and 15. The edge connectivity graph for this example is shown in Fig. 2.11b. Tracking any one edge and following the pointers would lead to the nearest edge to 0 which is 15.

2.4 CALCULATION OF BEZIER CURVE:

2.4.1 Choice of Bezier Points:

After forming the network, we next choose some points in the free space through which a Bezier Curve can be passed, the curve not fouling with any obstacle. In this section we will concentrate on how to choose Bezier Points and how the length of the curve is adjusted. A Bezier curve need not necessarily pass through the input points. Thus it is smooth in nature. The Bezier Points are chosen to be at the mid-point of the edges. The resulting curve thus would tend to be pulled towards the centre of the edge. Obviously the

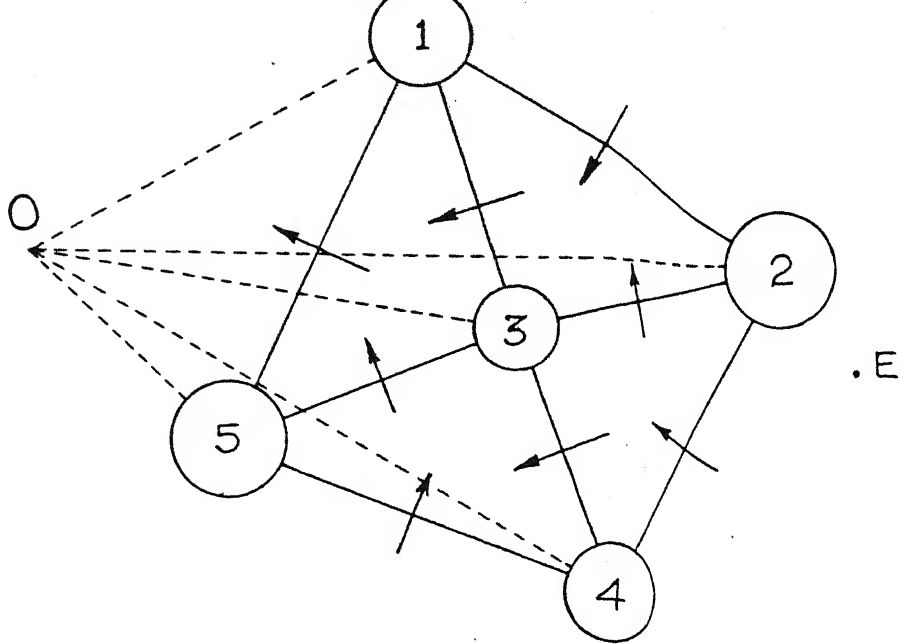


FIG 2.11 a : NETWORK OF 5 OBSTACLES

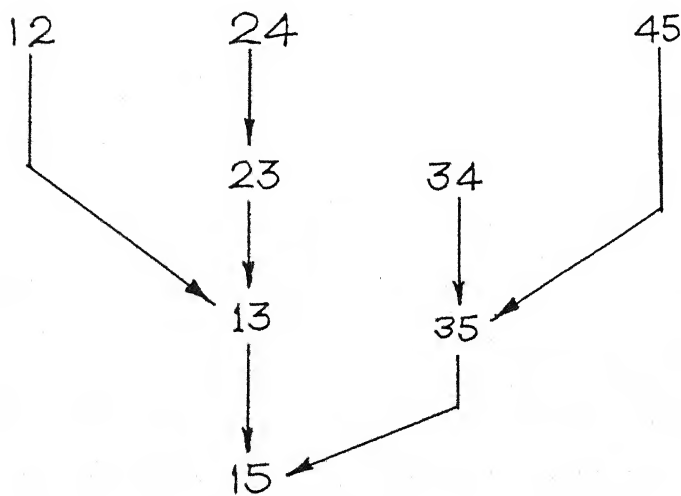


FIG 2.11 b : EDGE CONNECTIVITY GRAPH
OF NETWORK

curve is not guaranteed to pass through obstacle-free space, and in case the curve fouls with any obstacle, it is broken up at the fouling edge and joined smoothly at the mid-point. Other approximating functions like a spline fit or a B-spline would be computationally more involved and although smooth, they can have sharp turns which would be too stringent for modelling a robot manipulator.

To begin with, let us assume that we are given an end-effector position E and a suitable path for the manipulator has to be found. We are also given the "present manipulator state" and hence the "present first edge" as defined earlier in Section 2.2.3. If the state of the manipulator is 1, then the curve has to pass through edge $(1-1)1$ which is the present first edge only if E and O are not on the same side of this edge. Fig. 2.12a shows a workspace with 5 obstacles and the end-effector position is designated by E . The present manipulator state number is 3 which is also the end-effector natural state number. Then the present first edge is 23. The first Bezier Point is the mid-point (defined as R in Fig. 2.9) corresponding to edge 23. Now referring to the graph shown in Fig. 2.11b, we may follow the pointers. The next edges come out to be 13 and 15. Now construct a list which gives a rough path from position E to position O . This list is shown in Fig. 2.12b. The items in the lists are $EABCO$ where A , B and C are the mid-points of edges 23, 13 and 15 respectively. The points A , B and C have been stored in the sequence which occurs in the graph.

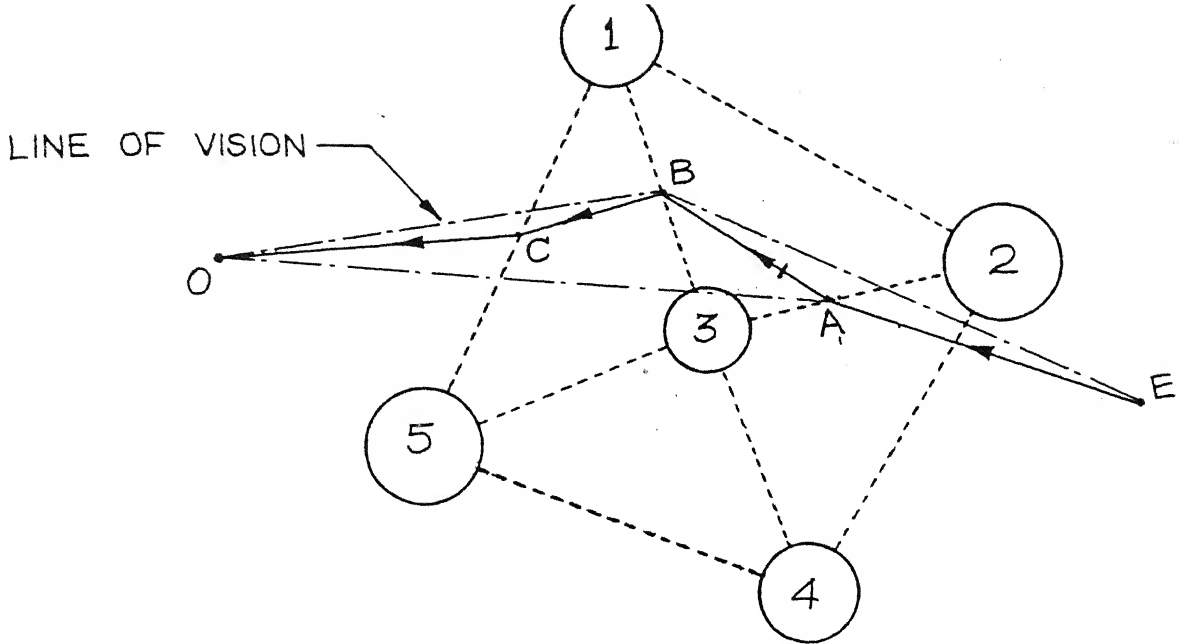


FIG 2.12 a: FOLLOWING POINTERS FROM E TO O

	23	13	15	
E	A	B	C	O
		✓		

EDGE
LIST
MARKERS

FIG 2.12 b: LIST REPRESENTATION WITH MARKER

A Bezier Curve is now to be calculated with these points. But before we do so, we like to ensure beforehand that the Bezier curve does not foul with any of the obstacles. A simple rule is followed to ascertain this condition. We assign markers (described in the following para) to some of these edges where a collision is likely, and break the Bezier curve into two or more parts, each joined smoothly at the suspicious looking edge.

Assigning markers is done in the following way. A list scanning is started from the base side and checked for visibility of O from each point of the list. Considering the example of Fig. 2.12, we find that O is visible from C and B but not from A. So we move down the list by one step, and put a marker at B. This is shown by a tick mark in Fig. 2.12b. This checking is continued up the list, but this time checking for visibility of B. It turns out that B is visible both from A and E. So there is only one marker in this case, which has been put at edge 13. Thus the number of Bezier curves that have to be passed are two, one from point E to point B and another from point B to point O. These curves are to be joined smoothly at edge 13.

So far the length of the curve has not been considered. Our requirement would be satisfied only when we have a curve of a definite length. However, we do have a set of points and if there are no markers, then a single Bezier Curve can be obtained. In case there are k markers, $(k+1)$ Bezier curves will

result whose continuity has to be maintained at the joining points. These curves when integrated, will give rise to a length, say L' . Now the rest of the length $L-L'$ (if > 0) has to be added to the last segment of the curve by having an additional point located at a suitable distance from O . Thus, the order of the Bezier curve is increased by one and the curve is expected to be elongated towards the base, so that the total calculated length lies close to the required length.

Integration of the curve is done by Modified Composite Simpson's Rule [8]. Coordinates of five intermediate points are calculated on the curve and integration is done. The procedure adopted for integration is outlined in the followings. As detailed in Appendix I a Bezier Curve is represented by

$$\vec{r} = \sum_{i=0}^n \vec{r}_i {}^nC_i (1-u)^{n-i} u^i \quad (2.13)$$

Differentiating it with respect to u yields

$$dx = \sum_{i=0}^n x_i {}^nC_i (1-u)^{n-i-1} u^{i-1} (i-1-u) du \quad (2.14a)$$

$$dy = \sum_{i=0}^n y_i {}^nC_i (1-u)^{n-i-1} u^{i-1} (i-1-u) du \quad (2.14b)$$

This expression will hold for $0 < u < 1$. But at the two extremities, the expression has to be modified to

$$\frac{dx}{du} \Big|_{u=0} = n(x_1 - x_0) \quad \frac{dy}{du} \Big|_{u=0} = n(y_1 - y_0) \quad (2.15a)$$

and

$$\frac{dx}{du} \Big|_{u=1} = n(x_n - x_{n-1}) \quad \frac{dy}{du} \Big|_{u=1} = n(y_n - y_{n-1}) \quad (2.15b)$$

Here m represents the number of points that have been taken on the Bezier curve for the purpose of integration. We have chosen m to be 5 for all cases. Integration can now be done by using the formula

$$l_j = \int_0^1 \sqrt{dx_j^2 + dy_j^2} \quad (2.16)$$

where the subscript j refers to the j^{th} Bezier curve. Thus l_j would represent the length of the j^{th} Bezier curve. The total length of all curves is found by summing up the length of all curves. Let the total calculated length be L' . Then

$$L' = \sum_{j=1}^k l_j \quad (2.17)$$

where k represents the total number of Bezier curves.

The problem of adjusting the final length of the curve by increasing the order by one, is now tackled. Finding a point placed at a suitable distance and a suitable angle is an involved job. An iterative procedure which has been found to have fast convergence is described next but it is to be noted that better methods can be easily found which are specific to the problem being solved. The present method has been chosen because of its computational simplicity. Referring to Fig. 2.13 we draw the bisector of the angle formed by line OC and edge 15. The bisector could be CX or CY. The choice of a bisector from CX and CY depends on which side the last link is more free to move. This is guided by the location of joint 2

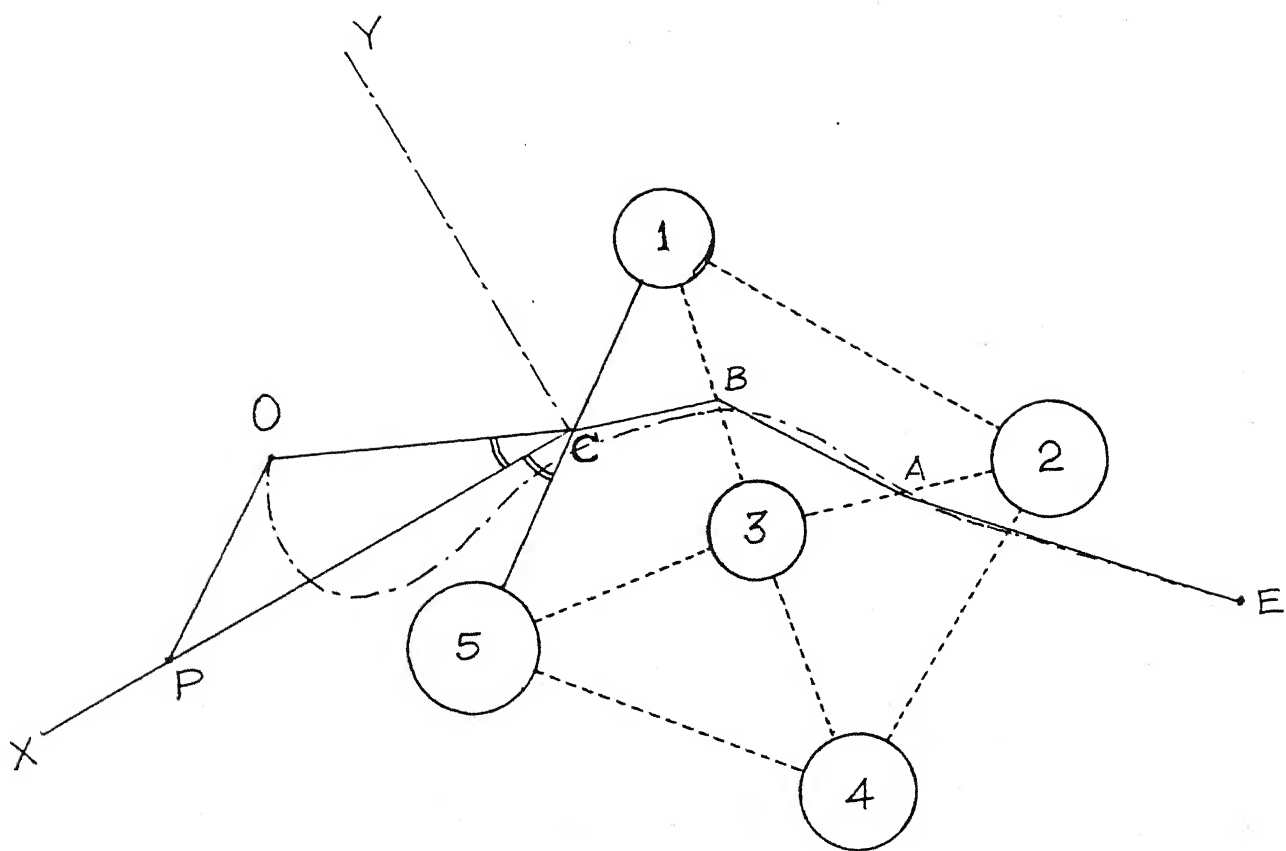


FIG 2.13 : ADJUSTMENT OF FINAL LENGTH

in the previous configuration. Let the required length of the last curve be L_r and the present calculated length be L_c . Then a point P is chosen on the line CX (or CY as the case may be) such that $\overline{PC} = \alpha(L_r - L_c)$ where α is any arbitrary constant (usually $0 < \alpha < 2.0$), and is to be determined iteratively. The new value of the integral, say L'' is compared with L_r and the value of α is updated by the formula $\alpha_{\text{new}} = \alpha_{\text{old}} \frac{L}{L_r}$. The calculation is repeated till $(L'' - L_r) \leq \delta$ where δ is some number representing the allowable error.

Upto now all the Bezier Points have been found and the next step is to construct the Bezier curves and find the joint coordinates as detailed earlier in Sec. 2.2.2. But we have so far not dealt with the first segment of the curve from the end-effector position E to the present first edge. This is taken up now in the following sub-section.

2.4.2 Determining Curve from End-Effector to Present First Edge:

We have earlier defined the term "present first edge" in Sec. 2.2.3. The present first edge is always an outside edge and is compared with the "natural edge" to assess the length of the first segment of the Bezier curve from E to the present first edge. Four cases may arise as has been shown later, and each has been dealt separately.

Before going into an analysis of the four cases, we like to reiterate a small point regarding "state change operation". This term has already been introduced in Sec. 2.2.3.

Going through a non-operational path would mean a deliberate state change operation, in order to have a most direct access to the end-effector position E . There are some cases where a state change operation is automatically attained. Fig. 2.14 shows two cases where the end-effector crosses a ray, namely ray 2. In the first case the transition is taking place from position E_1 to E_2 . The "end-effector natural state number" at position E_1 is 2 and let the "present manipulator state number" be 2. When the end-effector crosses over to E_2 the "end-effector natural state number" has changed to 3. But since the transition point on ray 2 is situated radially beyond the obstacle 2, the "present manipulator state number" has not changed and remains 2 provided the manipulator can access point E_2 from above the obstacle 2. In the second case, the transition is taking place from E_1' to E_2' . Let the "end-effector natural state number" and "present manipulator state number" at E_1' be the same as was in E_1 . Now since the transition point on ray 2 lies radially closer to 0 than obstacle 2, the "present manipulator state number" changes automatically with the "end-effector natural state number" and is equal to 3 at position E_2' .

Now we discuss the four cases that may arise when a path for a manipulator is to be found with a desired end-effector position and a given present manipulator state.

Case I: The "natural edge" coincides with the "present first edge" and E and 0 lie on opposite sides of the "present first edge".

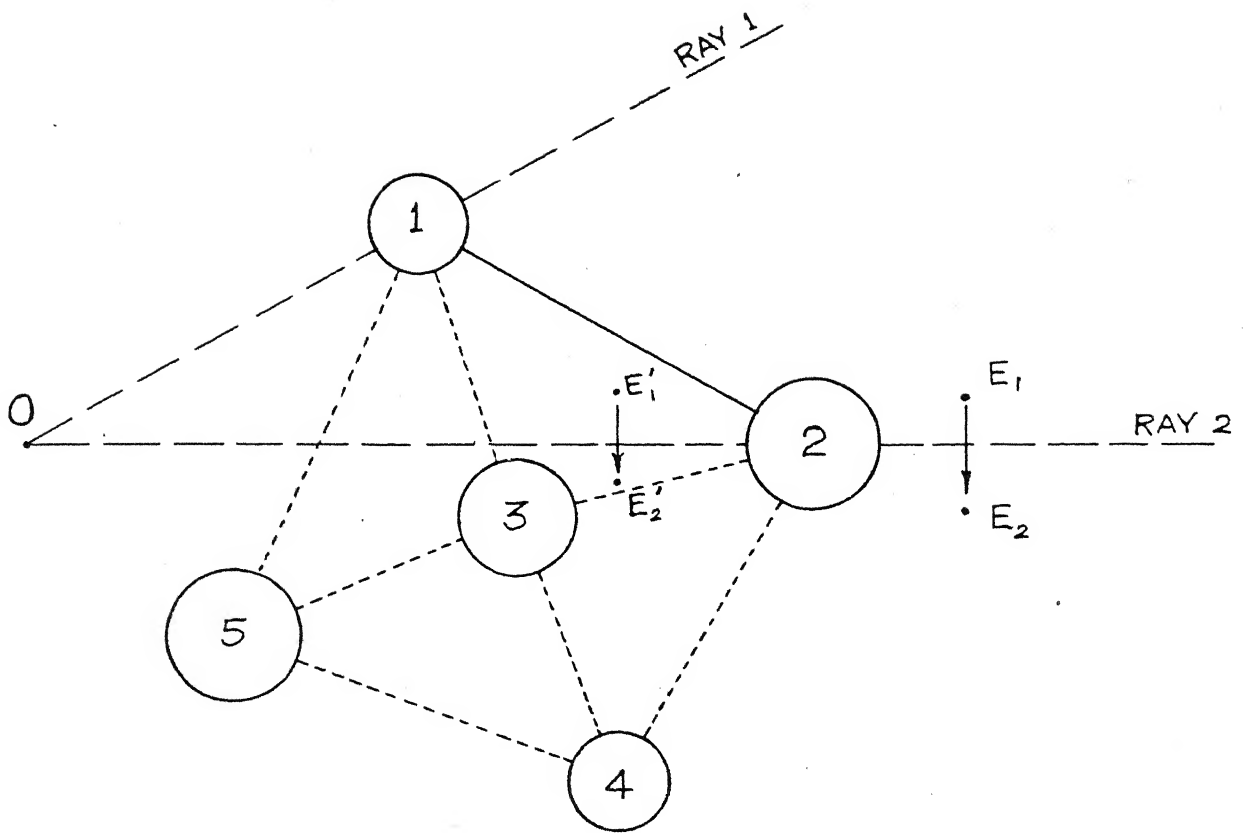


FIG 2.14 : FIGURE SHOWING CONDITIONS
OF A STATE CHANGE

This case is shown in Fig. 2.15(a). In this case the present first edge is obviously the natural edge. But certain precautions have to be taken if the line of vision of the mid-point of the present first edge, (P in Fig. 2.15a) from E is obstructed. If the line of sight is clear, then the point P is taken as the first Bezier point and the following edges are found by tracking pointers as explained earlier in Sec. 2.3.1. If EP collides with any obstacle then the next step is to place the Bezier point at the tail of the arrow of the present first edge. In Fig. 2.15a this point is F. If EF still collides with obstacle 2 or 3 (as in Fig. 2.15b), another point G is chosen such that it lies on the bisector of ray 2 and ray 3. The distance OG is adjusted so that it is the maximum of the outermost point of the two obstacles 2 and 3, i.e. $\max [O(2) + \rho_2, O(3) + \rho_3]$. After this exercise, a collision is now not possible.

Case II: The "natural edge" coincides with the "present first edge" and E and O lie on the same side of the "present first edge".

This case is depicted in Fig. 2.16 and is the simplest of all cases. The actual first edge which decides the first Bezier point, is now not the present first edge. The actual first edge is decided according to the following procedure. Referring to Fig. 2.16 the point E lies inside triangle O23. The present first edge in this case is 23. We follow pointers from edge 23 and reach edge 13. Now we check whether

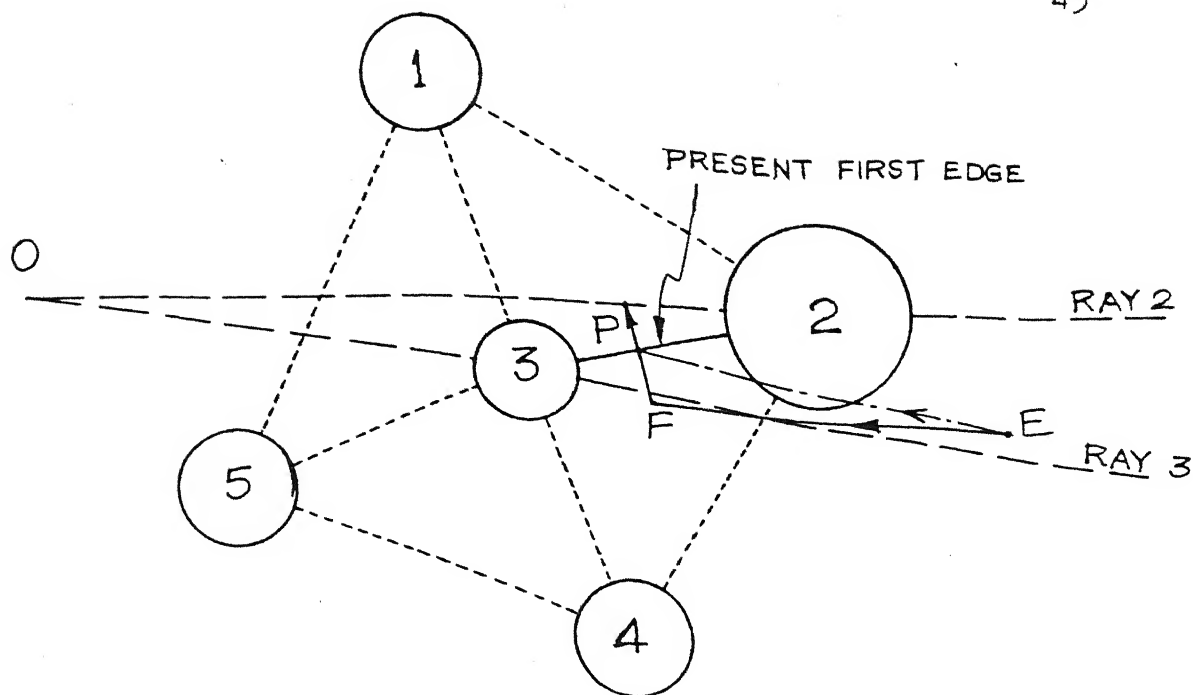


FIG 2.15 a: FIGURE DEPICTING CASE I

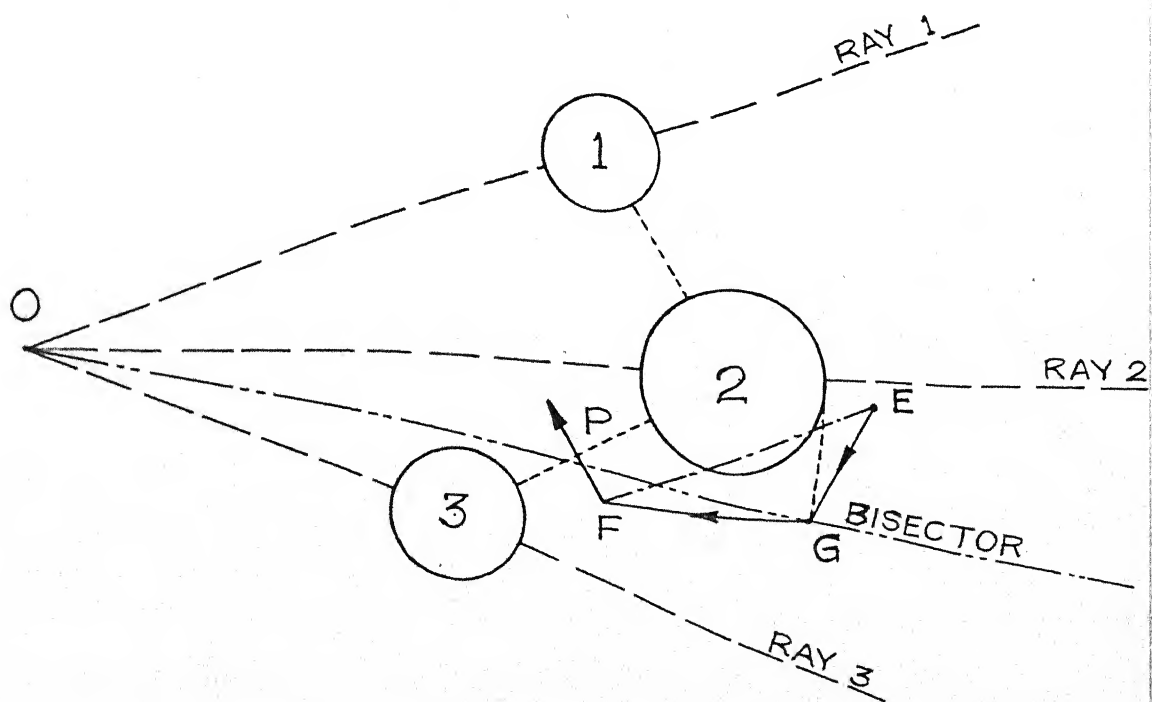


FIG 2.15 b: CASE I

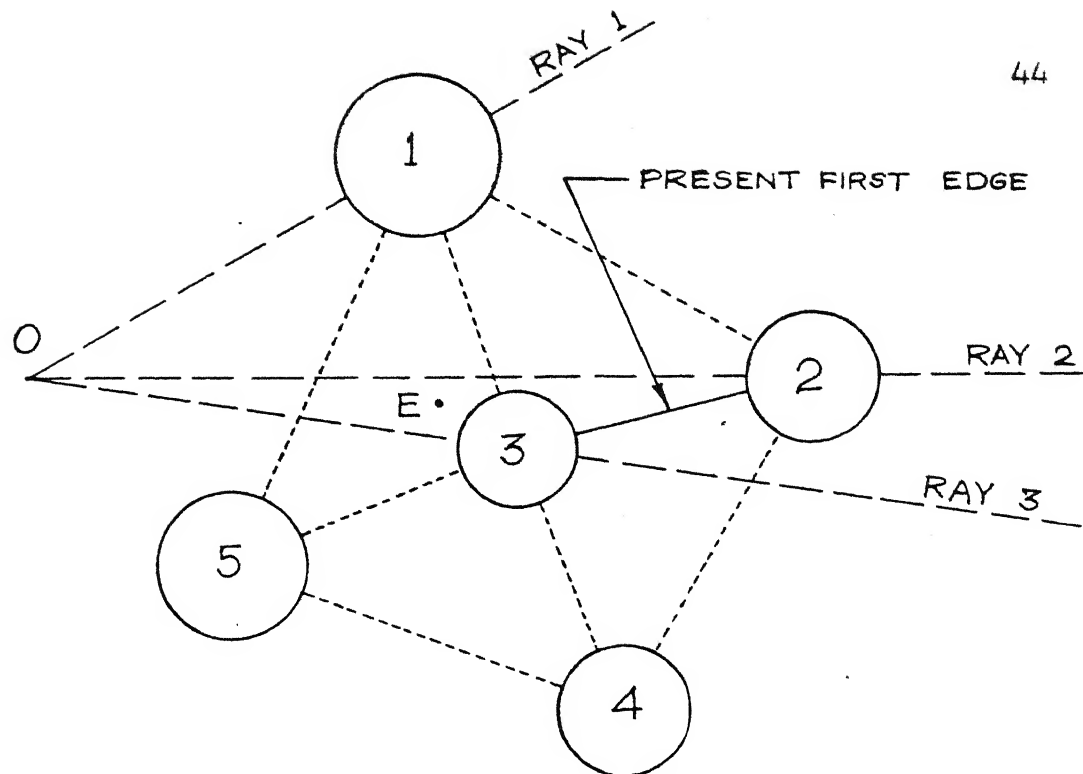


FIG 2.16 : FIGURE DEPICTING CASE II

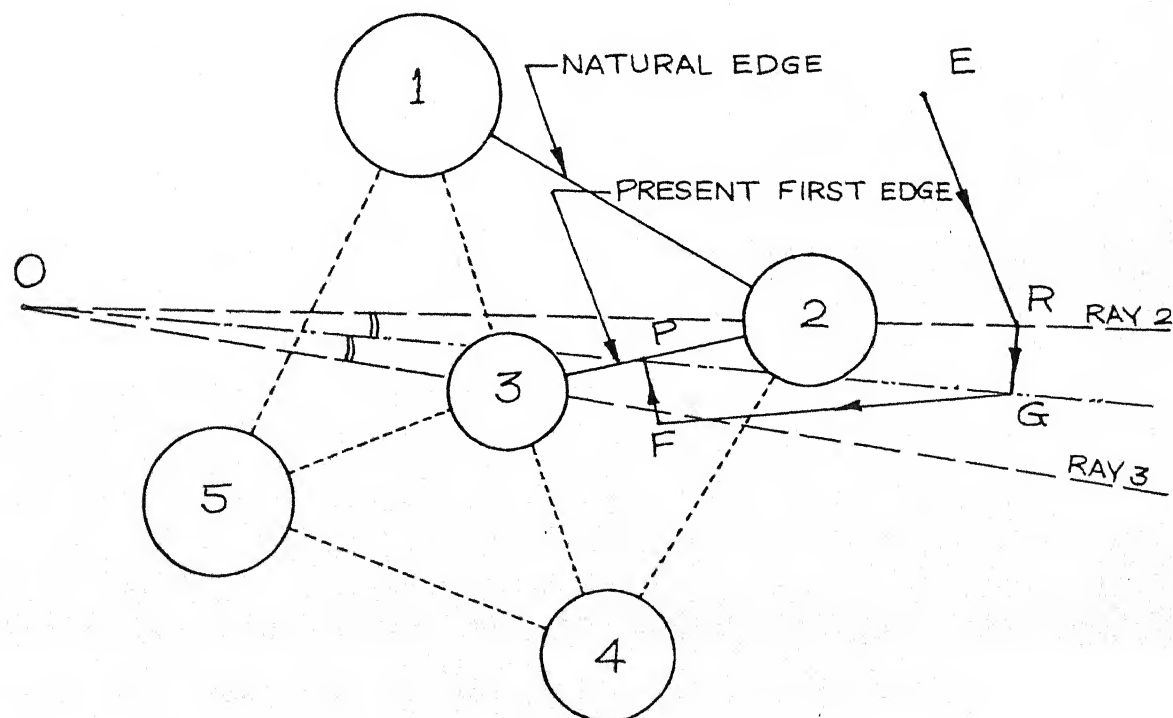


FIG 2.17 : FIGURE DEPICTING CASE III

point E lies inside triangle O13. In our case E is inside triangle O13. So we follow yet another pointer and reach edge 15. Now point E lies outside triangle O15. So the first Bezier Point is chosen on edge 15 and the rest of the procedure would be as detailed in Sec. 2.3.1. The situation where the point E lies outside the outer polygon (i.e. inside the triangle O15) is discussed in Chapter IV.

Case III: The "natural edge" and the "present first edge" do not match.

This case is shown in Fig. 2.17 and is a slightly complex situation. Many approaches are possible here and the aim is to circumvent all obstacles that may lie between the natural state (i.e. 2) and present state (i.e. 3), and reach the present first edge (i.e. edge 23), from the outer side. The first step is to check whether any obstacle is obstructing the line of sight from E to the mid-point of the present first edge 23. If the line of sight is free then we add that edge in the list and continue as in Case I. If the line of sight is being obstructed (as in Fig. 2.17) then some additional Bezier points have to be chosen on the ray 2. This point is represented as R in Fig. 2.17. The point R is chosen so that distance OR is equal to distance $O(i) + 3P_i$ where i is the obstacle number. In our case $i = 2$ since the obstacle to be circumvented is number 2. The next point is F as defined in Case I and finally the rest of the procedure is as shown in the earlier two cases.

Case IV: When no "present first edge" can be decided.

This case is depicted in Fig. 2.18. This case has been included to solve a robot configuration without considering the network at all. The "present manipulator state number" is 1 in this case. Any curve of length L from E to O is a valid choice as long as it does not interfere with any obstacle. A cubic curve has been chosen for the purpose and so two additional points are required. These points P and F (as shown in Fig. 2.18b) are chosen such that $OPFE$ forms a rectangle. We start with a point P placed so that OP is perpendicular to OE and P lies away from the centre C (as defined in Fig. 2.7a) of all the obstacles. The centre has been shown as C in Fig. 2.18b. A cubic Bezier curve is now passed from E to O and its length L' is found out by integration. The calculated length L' is now compared with the required length L . The corrected value of \overline{OP} is obtained by the formula $\overline{OP}_{\text{new}} = \overline{OP}_{\text{old}} \frac{L}{L'}$ and iteration is continued till $|L - L'| \leq \delta$ where δ is the allowable error.

2.5 PATH PLANNING OF END-EFFECTOR IN CASE OF COLLISION:

We have already discussed something about path planning in Sec. 2.2.3. There we had defined the terms "operational path" and "non-operational path" and had stated that a "non-operational path" would be required when a collision is imminent, implying that the "end-effector natural state" and the "present manipulator state" are different, or a point is too far away to be accessed in the "present manipulator state". In such cases, a non-

operational path would bring the "present manipulator state" to the "end-effector natural state" and the point can now be safely accessed. This section describes a procedure to find out the "non-operational path" in case of collision.

The "present manipulator state number" and the "end-effector natural state number" solely decide the non-operational path. Referring to Fig. 2.19 we find that the operational path has been defined by straight lines joining points E_1 , E_2 , and ^{so} on upto point E_4 . It so happens that at point E_2 , the next point E_2'' is no more accessible in present manipulator state number 2. The decision to retract and approach this point E_2'' from a new direction is now taken and the number of obstacles to be circumvented is to be found out. Suppose the "present manipulator state number" is p and the "end-effector natural state number" of the next end-effector position E_2'' is q . Then the number of obstacles to be circumvented are $|q-p|$. The obstacles being circumvented will have the numbers $p, p+1, \dots, q-1$ (if $q > p$). In Fig. 2.19 the present manipulator state number is 2 and the end-effector natural state number at position E_2 is 4. Thus $(4-2) = 2$ obstacles have to be circumvented, namely obstacles 2 and 3.

The path to be followed in circling around the obstacles is quite easy to find once the obstacles to be circumvented are identified. Keeping a safe margin ϵ around the obstacles we approach the first obstacle tangentially from the present end-effector position E_2' . This gives rise to the path $E_2' S_1$.

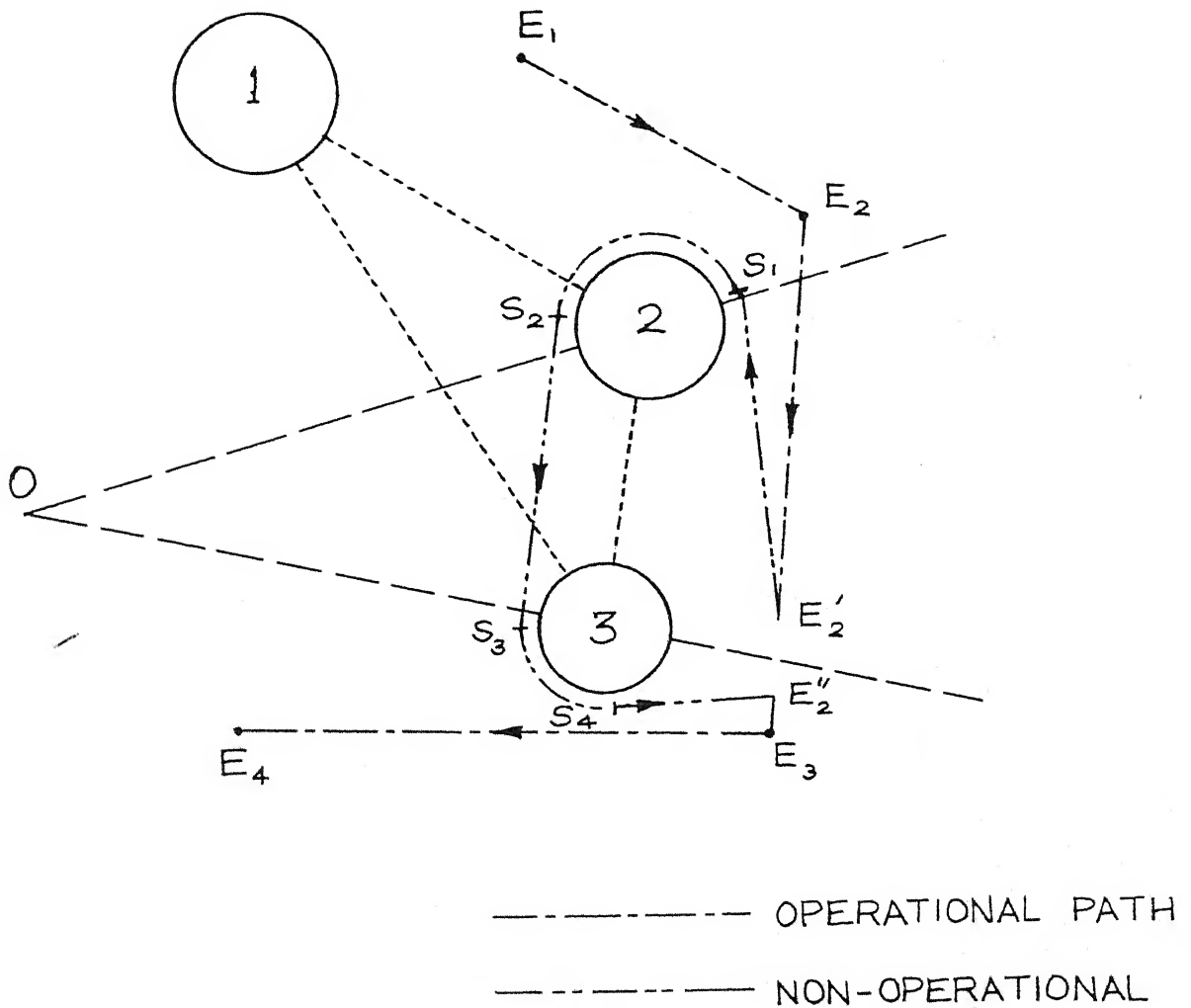


FIG 2.19: CIRCUMVENTING OBSTACLES
IN CASE OF COLLISION

A circular path $S_1 S_2$ is next traced keeping the same radius till a tangential path $S_2 S_3$ to the next obstacle is found. This process of tracing a path along the outer periphery of each consecutive obstacle is continued till the last obstacle is reached ($S_3 S_4$). From there a tangential path $S_4 E_2''$ is now followed to the next end-effector position of the operational path which is E_2'' . In this process the "present manipulator state number" keeps changing each time we cross a ray, so that when we are at the new position E_2'' , the "present manipulator state number" is equal to the "end-effector natural state number."

If the point is still not accessible then the point is really inaccessible because of insufficient link lengths or limitations on joint rotation. In such a case the job may be aborted or the next end-effector position, is tried.

CHAPTER III

FORMULATION AND IMPLEMENTATION OF ALGORITHM

3.1 INTRODUCTION:

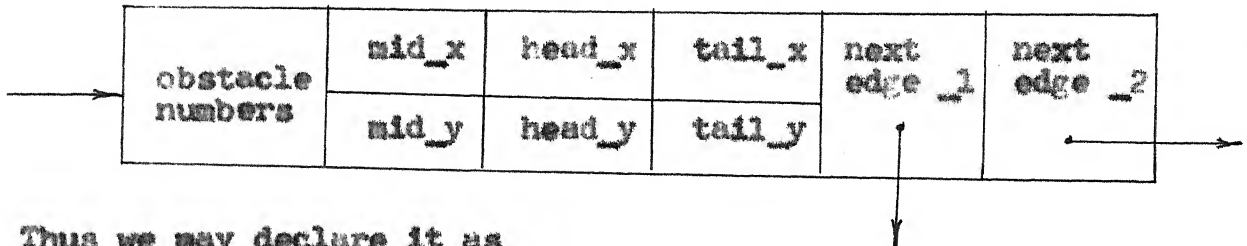
In the previous chapter a method was discussed to determine the joint coordinates of a robot with redundant links. The method involves firstly representing the workspace as a network whose edge connectivity is established in the form of a graph. This network is used to find out a curve from the end-effector position E to the base O. The shape of the curve is chosen so that it avoids collision with any of the obstacles lying in the workspace. This curve is now used to find out the joint coordinates of the robot. All through, only planar cases are taken up, where the obstacles are all circular and the manipulator is free to move in this plane only.

In this chapter we propose to give some of the implementation details and the algorithm description.

3.2 NETWORK REPRESENTATION:

Representation of the network requires an efficient data structure. The data structure establishes a relationship between the adjacent edges. The record also stores the mid-points of the edges, the coordinates of the head and tail of the arrow attached to the edge, as well as the obstacle numbers

which lie at its ends. In figurative form it may be represented as:



Thus we may declare it as

type

(3.1)

```

obstacle_points = array [1..2] of integer;
x_and_y_coordinates = array [1..2] of real;
pointer_to_edge = ^ edge;
edge = record

```

```

    obstacles      : obstacle_points;
    mid_point      : x_and_y_coordinates;
    head           : x_and_y_coordinates;
    tail           : x_and_y_coordinates;
    next_edge_1    : pointer_to_edge;
    next_edge_2    : pointer_to_edge

```

```

end;

```

The formation of the network requires firstly forming an outer polygon around all the given obstacles. The obstacles are now selected with three in a group and possible non-overlapping triangles are identified. Lastly, the edges of these triangles are chosen one at a time and the relationship between the adjacent edges are established.

Thus forming the network is done by

procedure form_network; (3.2)

```
begin
    form_outer_polygon;
    form_triangles;
    identify_and_link_edges;
end;
```

Each of these subprocedures are described in the following these subsections.

3.2.1 Forming Outer Polygon:

The first procedure, namely form_outer_polygon involves first describing a circle which is large enough to enclose all obstacles. The obstacles are then viewed from a point which lies on the circumference of this circle and the nearest obstacle stored in stack. The viewing point is changed in small steps along the circumference and the process is terminated when one revolution is complete (Refer to Sec. 2.2.4). In algorithmic form it is thus written as

```
procedure form_outside_polygon(n:integer;x,y:array[1..n]
                                of real); (3.3)
```

```
begin
    mid_x := (max(x[1] to x[n]) + min(x[1] to x[n]))/2;
    mid_y := (max(y[1] to y[n]) + min(y[1] to y[n]))/2;
    radius := max ((max (y[1] to y[n])-mid_y),
                   (max (x[1] to x[n])-mid_x));
    circular_list_pointer := 0;
    read (starting_angle, max_intervals);
    angle_increment := 360/max_intervals;
    for k := 1 to max_intervals do
```

```

begin
    view_x := mid_x + radius * cos (starting_
        angle + k * angle_increment);
    view_y := mid_y + radius * sin (starting_
        angle + k * angle_increment);
    find nearest point from (view_x, view_y);
    if last_element ≠ present_element then
        begin
            circular_list_pointer := circular_
                list_pointer + 1;
            element [list_pointer] := present_
                point
        end;
    end;
end;

```

3.2.2 Forming Triangles:

Having formed the outside polygon surrounding all obstacles, the next step is to divide the outside polygon into a set of non-overlapping triangles. The centres of the obstacles form the vertices of the triangle. The outer polygon which has been formed will help us to identify some of the edges of these triangles.

The strategy adopted for forming triangles out of the outer polygon has been discussed in Sec. 2.2.4. The present work has however not dealt with any case where the number of left out obstacles exceed one. But these cases can easily be incorporated by using the procedure mentioned in Sec. 2.2.4.

In algorithmic form we may thus state

```

procedure form_triangles;                                     (3.4)
begin
  case (total_number_of_obstacles - circular_list_
        pointer) of
    0: begin
        repeat
          for each element of the list do
            begin
              take three elements from list
              and form trios;
            end;
            present_list := new_list;
          until present_list_is_empty;
        end;
    1: begin
        repeat
          take two elements from list and
          combine with element not in list;
        until list_is_empty;
      end;
    others: begin
        form_outside_polygon (n-circular_list_
                              pointer, x,y);
        join_inner_outer_polygon;
        form_triangles;
      end;
  end;
end;

```

3.2.3 Establishing Edge Connectivity:

The last step in forming the network is to identify the edges of the triangles, choosing them one at a time. The arrow attached to the edge is next drawn and then connectivity of the edges is established. As described in Sec. 2.3.1, three cases may arise and these have been dealt separately in the following procedure.

```

procedure identify_and_link_edges (triangle_set: set of
  triangles; max_triangles: integer);                       (3.5)
begin
  for each triangle j := 1 to max_triangles do
    begin
      identify_all_edges_of_triangle(j);
      for each edge i := 1 to 3 do

```

```

begin
  assign direction of arrow;
  if arrowhead of edge (i) is inside
    triangle (j) then
    arrow_inside := true
  else
    arrow_inside := false;
end;
for each edge i := 1 to 3 do
begin
  if arrow_inside (i) then
    begin
      if the other two arrows, u and
        v are pointing outside then
        begin
          edge(i).next_edge_1 :=
            (pointer_to_edge_u);
          edge(i).next_edge_2 :=
            (pointer_to_edge_v);
        end
      else
        if only one of the other two
          arrows are pointing
            outside then
            begin
              edge(i).next_edge_1 :=
                pointer to that edge
                whose tail is inside
              edge(i).next_edge_2 :=
                nil;
            end
          else
            begin
              edge(i).next_edge_1 :=
                nil;
              edge(i).next_edge_2 :=
                nil;
            end;
        end;
      end;
    end;
  end;
end;
end;
end;
end;

```

The network is now ready. During execution, the pointers will act as a guideway from the end-effector position to the base O. Whatever has been discussed so far is done before the actual execution starts. In case of static obstacles, the network will not change and the same network will be used to find out the configuration for each end-effector position. This would thus form part of the pre-processing time. In a dynamic environment, the network will change, but now

edges would be allowed to overlap. The only difference that may occur is in the position of the obstacles and the location of the mid-point and coordinates of the arrows.

3.3 FINDING MANIPULATOR CONFIGURATION:

The network which has been generated by procedure `form_network` is used to find out the manipulator configuration for each end effector position. This task is thus broken up into two sub-tasks. Thus overall control of the task is done by

```
begin { Main Program }                                     (3.6)
  repeat
    find_next_EE_position;
    find_manipulator_configuration;
  until end_of_task;
end;
```

These two sub-tasks are discussed in the following sub-sections.

3.3.1 Finding Next End-Effector Position:

The next end-effector position depends on whether it is tracking an operational path or a non-operational path. The decision that a non-operational path is to be tracked is discussed later in Sec. 3.3.2.6. Assuming that the information about the nature of the path is already known, the outlines for determining the end-effector position are,

```
procedure find_next_EE_position;                             (3.7)
begin
  if EE_is_in_operational_path then
    begin
      find the next point by straight line
      interpolation;
    end
  else
    begin
      choose the next point on the non-operational
      path already calculated;
    end;
end;
```



```

: E is towards O from present_first_
edge:
begin
  present_edge := present_
    first_edge;
  while line OE does not
    intersect the present_first_
    edge do
    present_edge := nearest to
      base (present_edge.next_
        edge_1, present_edge.
        next_edge_2);
  end;
endcase;
build_stack_of_edges;
assign_markers_to_edges;
select_Bezier_Points;
find_joint_coordinates;
end;
: EE_natural_state / present_manipulator_state
begin
  find the number of obstacles to be
    crossed;
  find a path from E to present_first_
    edge;
  present_edge := present_first_edge;
  build_stack_of_edges;
  assign_markers_to_edges;
  select_Bezier_Points;
  find_joint_coordinates;
end;
: Present_natural_edge cannot be decided:
begin
  fit a curve of order three without
    considering the network;
  find_joint_coordinates;
end;
endcase;
check_collision_cases;
end;
end;

```

The subroutines used in the above procedure are discussed now with a short description.

3.3.2.1 Finding End-Effector Natural State:

The end-effector natural state is just a number depending on the end-effector position. This number is used for comparison purposes with the present manipulator state and the path to be

taken by the Bezier Curve is decided upon. In physical terms, this would represent the most natural access to the end-effector position from base 0. The procedure for finding the end-effector natural state is as follows.

```

procedure find_EE_natural_state;                                (3.10)
begin
  natural_state := 0;
  find_end_effector_angle;
  repeat
    natural_state := natural_state + 1;
  until end_effector_angle > obstacle_angle[natural_state];
end;

```

3.3.2.2 Building Stack of Edges:

The stack, which contains the index of the edges, gives the rough path between the base 0 and end-effector E. This path is smoothened later by using Bezier Curve. The stack is build by essentially following pointers from the present_first_edge. Hence, the procedure can be outlined as

```

procedure build_stack_of_edges;                                (3.11)
begin
  stack_top := 1;
  element[stack_top] := present_edge;
  while present_edge.next_edge_1 ≠ nil do
    begin
      stack_top := stack_top + 1;
      present_edge := nearest_to_base(present_edge.
        1, present_edge.next_edge_2);
      element[stack_top] := present_edge;
    end;
  end;
end;

```

3.3.2.3 Assigning Markers to the Edges:

Markers are assigned to one or more edges that are present in the stack. This is done to break the curve into one or more parts, to ensure beforehand that potentially dangerous obstacles do not collide with the curve (see Sec. 2.4.1). Even though

this does not guarantee a collision free path, it is nevertheless useful to avoid recalculation in some cases where a collision is likely.

Markers are put at some edges by checking that the line of sight from the start of the curve to the end is free. The complete description was given in Sec. 2.4.1. In algorithmic form we have

```

procedure assign_markers_to_edges;                                (3.12)
begin
    viewing_point := base_coordinates;
    stack_pointer := stack_top;
    repeat
        while element [stack_pointer].mid_point is
            visible from viewing_point and not end_of_
                stack do
            stack_pointer := stack_pointer - 1;
        if not end_of_stack then
            begin
                stack_pointer := stack_pointer + 1;
                assign marker at element [stack_pointer];
                viewing_point := element [stack_pointer];
            end;
        until end_of_stack;
    end;
end;

```

3.3.2.4 Selecting Bezier Points:

The stack alongwith markers are now used to select Bezier Points which will establish the Bezier curves. When a marker is present at an edge, the curve is broken up at that edge and made to pass through the mid-point of that edge. The two curves are joined so that continuity is maintained at the joining point. Selecting Bezier Points is done by

```

procedure find_joint_coordinates;                                (3.14)
begin
  i := number_of_links;
  present_x := end_effector_point_x;
  present_y := end_effector_point_y;
  repeat
    choose the appropriate Bezier Curve;
    calculate u for required point;
    find_point_on_Bezier_curve (u, curve_number, x, y);
    from (present_x, present_y) join a straight
                                         line to (x, y);
    cut a length link_length (l) on this line at (x', y');
    present_x := x';
    present_y := y';
    joint_coordinate(i) := difference in slope of
                                         adjacent links;
    i = i-1;
  until i = 2;
  adjust_last_two_links (x0, y0, present_x, present_y,
                        a1, a2, x1, y1);
end;

```

The above procedure involves two procedures, namely `find_point_on_Bezier_curve` and `adjust_last_two_links`. These are now described in the followings.

Finding a point on a Bezier curve is a simple step. Once the curve and Bezier Points are known. Thus we have

```

procedure find_point_on_Bezier_Curve (u: real; curve_number:
                                     integer; var x, y: real);    (3.15)
var max_points : integer;
begin
  x := 0;
  y := 0;
  max_points := number_of_Bezier_Points [curve_number];
  for i = 1 to max_points do
    begin
      C := combination (max_points, i);
      m := (1-u)n x (max_points-1) x u x n-1 1;
      x := x + Bezier_point_x [i, curve_number]
            x C x m;
      y := y + Bezier_point_y [i, curve_number]
            x C x m;
    end;
  end;
end;

```

The Bezier curve cannot be used to find the last two links.

The position taken by joint 2 is calculated, and among the two

possibilities that may result, the one which is nearer to the previous joint position will be preferred. A description without the logic statements is now given below.

```

procedure adjust_last_two_links (x0,y0,x2,y2,a1,a2:real;
                                var x1,x2:real);      (3.16)
var K1,K2: real;
begin
  K1:= ((x2-x0)2+(y2-y0)2+a12-a22)/(2 * a1);
  alpha:= ATAN2 ((x2-x0),(y2-y0));
  K2 := K1/SQRT ((x2-x0)2 + (y2-y0)2);
  theta := alpha + COSINV (K2);
  x1 := x0 + a1 * COS (theta);
  y1 := y0 + a1 * SIN (theta);
end;
```

With this the description of find_manipulator_configuration (algorithm 3.9) needs only check_collision_cases to be described. The resulting configuration will approximately take the shape taken by the Bezier curve. But collision between the obstacles and the calculated link positions are still likely. So the next step is to check whether a collision is taking place, and if so, to incorporate corrective measures that tend to eliminate collisions. The next sub-section deals with this problem.

3.3.2.6 Dealing with Collision Cases

The procedure check_collision_cases is in two parts. It first checks out collision cases if any, and then takes corrective measures.

```

procedure find_collision_cases;      (3.17)
begin
  for i := 1 to number_of_links do
    for j := 1 to number_of_obstacles do
      check whether link [i] collides with
                                obstacle[j];
    end;
```


If a collision is occurring then a procedure is called to modify the manipulator configuration. This procedure assigns an additional marker at a suitable edge and recalculates the Bezier points followed by a recalculation of the joint coordinates. The procedure to modify the manipulator configuration can be stated as

```

procedure modify_manipulator_configuration;      (3.18)
begin
    put a marker at edge which is connected to the
        fouling obstacle and whose mid_point is nearest;
    select Bezier_Points;
    find_joint_coordinates;
end;
```

The above two procedures are added together and combined in a single procedure check-collision-cases. If the modification done by modify_manipulator_configuration does not give a collision free path then a state change is necessary. So the non-operational path would be calculated and the end-effector will be directed to move on this non-operational path. We may now enumerate these steps in the following procedure.

```

procedure check_collision_cases;                  (3.19)
begin
    find_collision_cases;
    if collision_is_occurring then
        begin
            modify_manipulator_configuration;
            find_collision_cases;
            if collision_is_occurring then
                begin
                    find_non_operational_path;
                    direct the manipulator on the
                        non-operational path;
                end;
            end;
        end;
end;
```

With this procedure the description of find_manipulator_configuration is complete. This procedure is self-sufficient to indicate

the necessity of a state change. After the state-change operation is over, the end-effector will resume its path on the operational-path.

3.4 SOME ADDITIONAL PROCEDURES:

This section describes two procedures. The first procedure checks whether a line cuts a circle and is used to check whether a link is fouling with any obstacle (Sec. 3.3.2.6). The second procedure checks whether a point lies inside a triangle and is used to consider cases of Sec. 2.4.2. Here the algorithms have been written as functions with boolean results.

Referring to Fig. 3.1, the first function is

```
function line_cuts_circle ( $x_b, y_b, r_b, x_1, y_1, x_2, y_2$ :real):
                                boolean;      (3.20)
begin
    from centre ( $x_b, y_b$ ) find perpendicular distance d
        to straight line ( $x_1, y_1$ ) and ( $x_2, y_2$ );
    if  $AB^2 < r_b^2$  or  $AC^2 < r_b^2$  then line_cuts_circle := true
    else
        if  $d^2 > r_b^2$  then line_cuts_circle := false
        else
            if angle ABC is acute then
                line_cuts_circle := false
            else
                line_cuts_circle := true;
    end;
```

The second procedure is described now (Fig. 3.2).

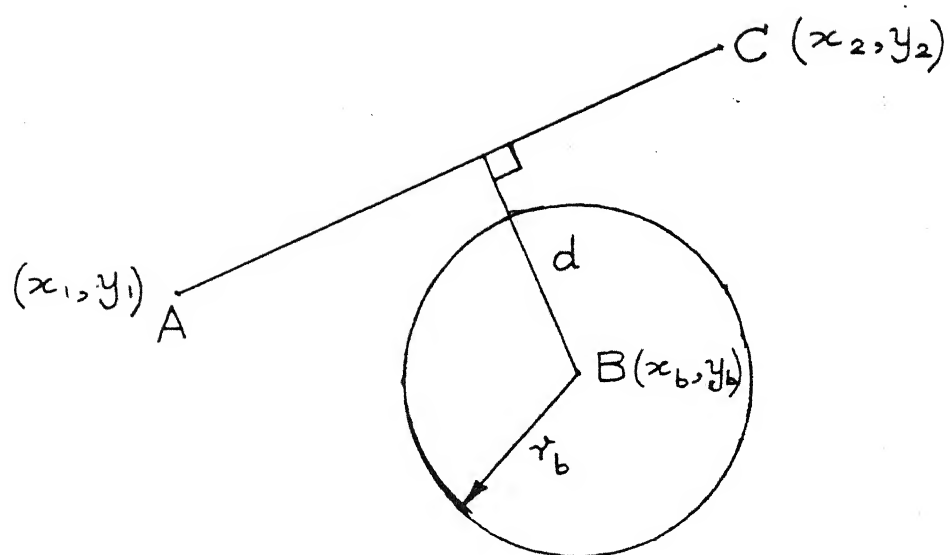


FIG 3.1 LINE_CUTS_CIRCLE

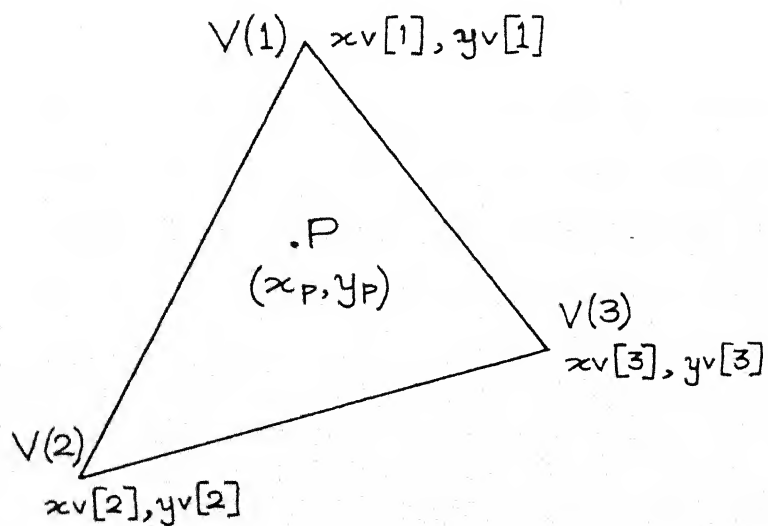


FIG 3.2 POINT_INSIDE_TRIANGLE

```

function point_inside_triangle (xV,yV: array [1..3] of
                                real; xP,yP: real): boolean;      (3.21)
var
    apart: array [1..3] of boolean;
begin
    for each vertex i := 1 to 3 do
        begin
            join V(i) to P
            if the other two vertices lie on opposite
               side of line V(i)P then
                apart [i] := true
            else
                apart [i] := false;
        end;
    if for any two vertices j and k
        apart [j] = true and apart [k] = true then
        point_inside_triangle := true
    else
        point_inside_triangle = false;
end;

```

The above algorithms have been written in a Pascal-like language and conversion to any structured language is easy. The actual program codes have, however, been written in BASIC on an Apple 2e micro-computer and are given in Appendix III.

CHAPTER IV

RESULTS AND DISCUSSIONS

4.1 INTRODUCTION:

Three different problems are presented in this chapter. In all three cases, the end-effector has to access the interior walls of different objects. The first two problems are such that the entire operational path is covered in one sweep, whereas, in the last problem the object is such that, nearly halfway through the operational path, the manipulator has to retract and a non-operational path has to be planned.

The algorithm has been implemented on an Apple IIe micro-computer. The source listing is given in Appendix III. The program is of an interactive nature and high resolution screen pages are used to display the object and the manipulator on the screen. The description of this interactive programme is presented in Appendix II.

The present algorithm does not account for solid edges. Therefore the objects have to be suitably modelled by placing "pseudo-obstacles" in the workspace. Selection of the obstacles needs a little care and some guidelines are given below.

- 1) If the object is symmetric, it is better to place one obstacle at the line of symmetry rather than two flanked side by side, so that the network is symmetric.

- 2) Chose the obstacles in such a way so that the mid-points of the edges lie inside the working space of the robot.
- 3) Space the obstacles in such a way that the end-effector positions lie inside the outer polygon formed by the obstacles or at least not very far away outside the polygon. This ensures sufficient Bezier points for the algorithm.
- 4) Two obstacles touching each other should be replaced by one large obstacle enveloping both.
- 5) If possible, try to keep the base 0 outside the outer polygon formed by the obstacles.

4.2 OBJECT WITH THREE INTERIOR EDGES:

The object is shown in Fig. 4.1 and is the same as used in Ref. [4]. The end-effector has to track a path which is specified by the straight lines joining the points E_1, E_2, E_3, E_4, E_5 and E_6 . Thus, the operational path is shown in Fig. 4.1 by chain dotted lines. This path is tracked by taking thirty four nearly equally spaced end-effector positions on the path.

The object is modelled by placing four obstacles in the workspace as shown in Fig. 4.2. Obstacles 1 and 3 have been placed at the opening of the cavity to avoid a collision. Obstacles 2 and 4 have been placed at either side of edge 13 to include the space inside the cavity as well as the space outside it, into the workspace. The triangles 123 and 134 is the set selected to form the network. The set of triangles 124 and 234 is rejected as the triangle 234 is too flat and this will introduce difficulties in following the arrow of edge 24. Fig. 4.2 also shows the arrows, with directions assigned to each edge

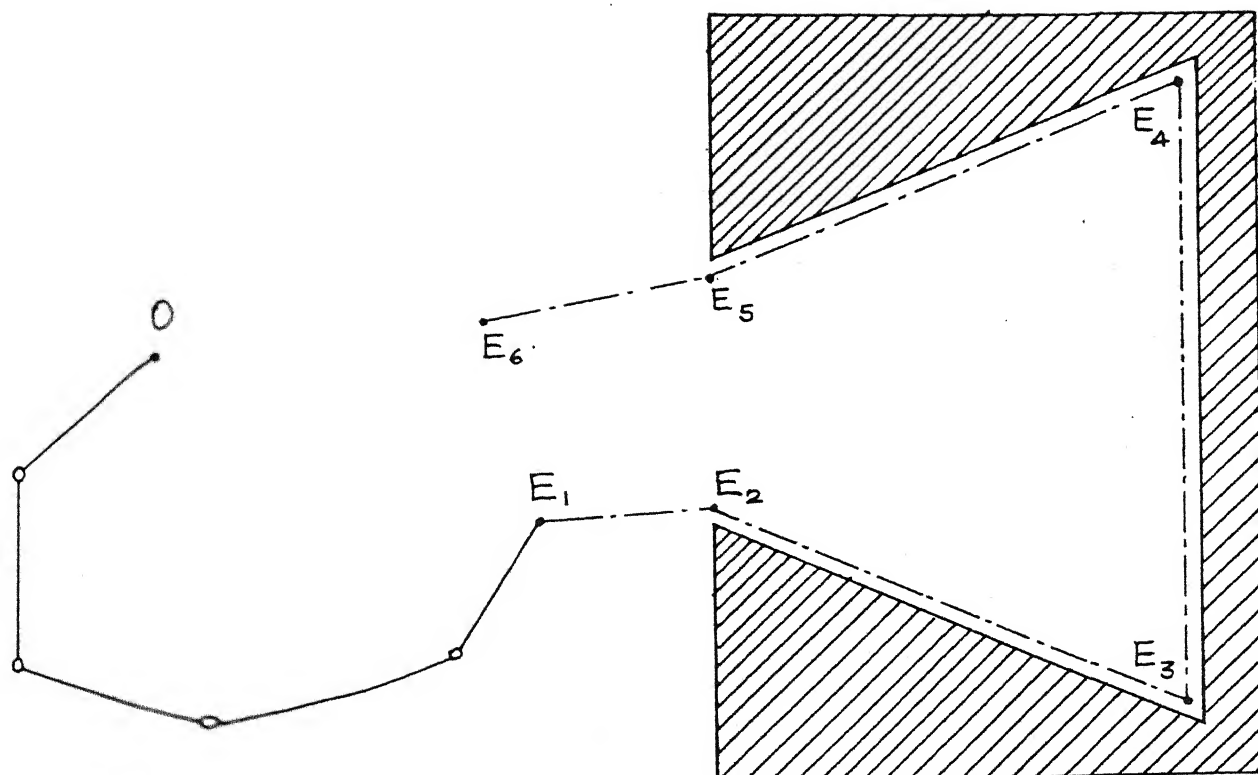


FIG 4.1 : PROBLEM 1 : DEFINITION

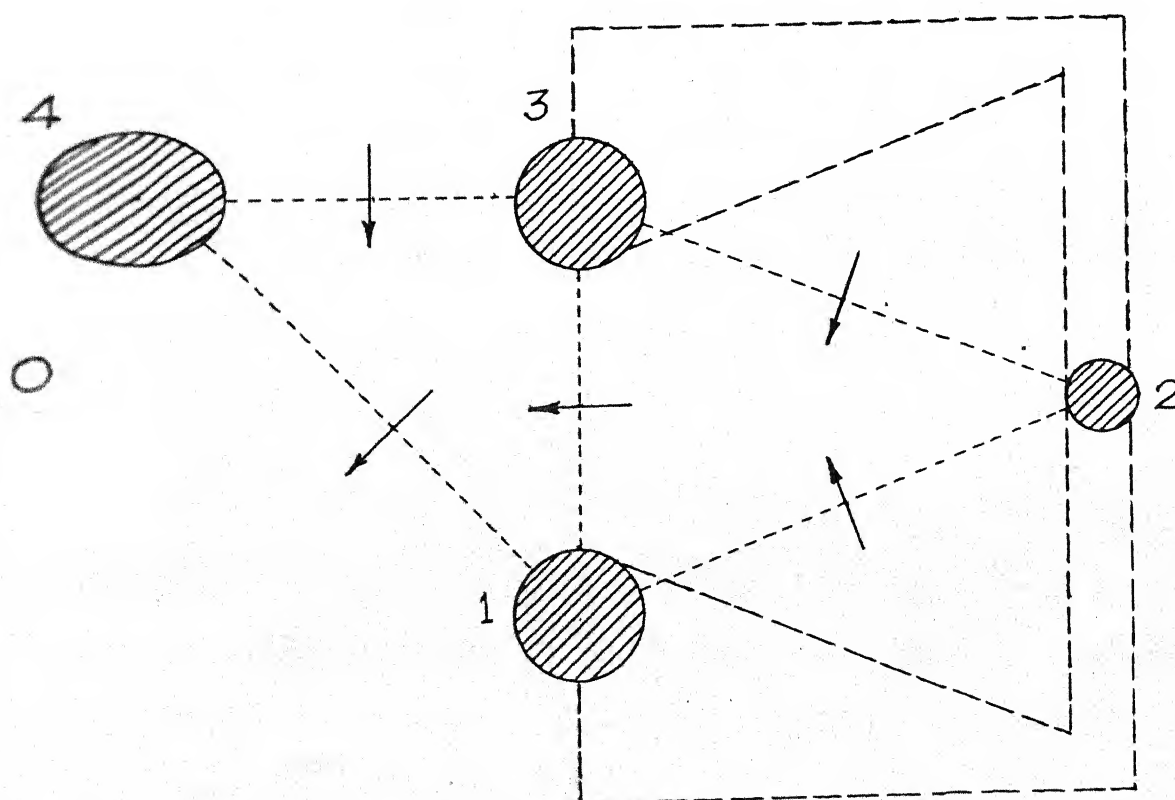


FIG 4.2 : MODELLING OF SPACE

according to the rules layed down in Sec. 2.3.1.

This problem was solved by considering manipulators with 3,4,5,6 and 8 links. In each case, the sum of lengths of all the links is kept same and is equal to 200 units. The three link and the four link manipulators were found to successfully perform the task. The results are presented only for five, six, and eight link manipulators.

Two cases of five link manipulators are considered. In the first case, all the link lengths are equal and are 40 units. The configurations for positions 3, 6 and 14 are shown in Fig. 4.3 and positions 18, 29 and 34 are shown in Fig. 4.4. Fig. 4.5 shows superimposed alternate configurations for the entire task.

Fig. 4.6 shows the Bezier Points E, A, B, C and D for the position number 29 of the end-effector. As is evident from this figure, when E crosses the edge 23, the second Bezier point abruptly changes from A to B. This causes large rotations of the joints, particularly of the joints located towards the end-effector. A method of smoothening this transition at the end-effector side of the manipulator has been used. For this, consider the list represented in Fig. 4.6. The nearest edge to E is 23 whose mid-point has been represented as point A. The acute angle between the line of arrow at point A and line EA is calculated. This angle is shown as ϕ in the figure. Then the next Bezier point is chosen at A' instead of A where A' lies on line AB and

$$A'E = AE \cos \phi \quad (4.1)$$

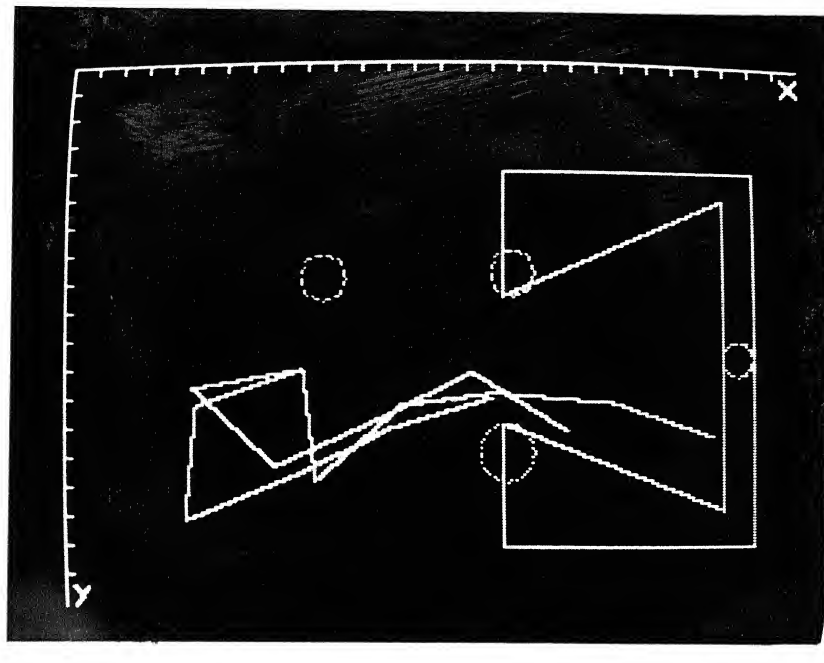


FIG 4.3 : 5-LINK MANIPULATOR
POSITIONS 3, 6 & 14

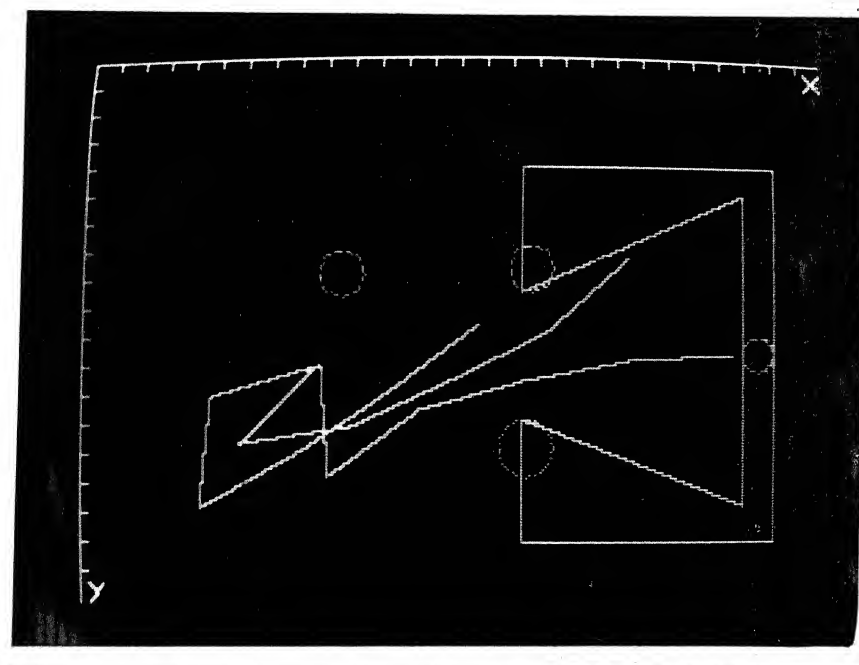


FIG 4.4 : 5-LINK MANIPULATOR
POSITIONS 18, 29 & 34

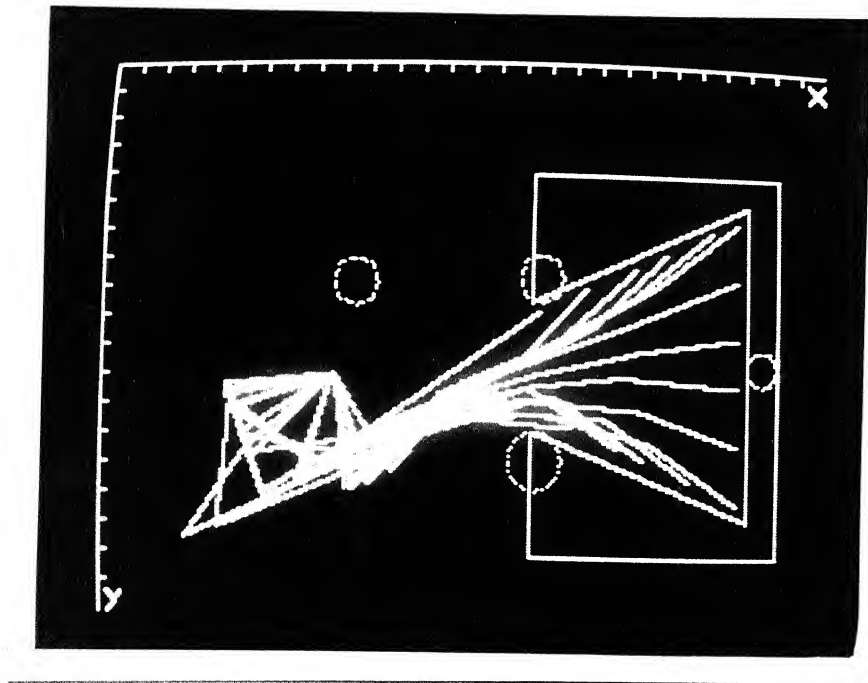
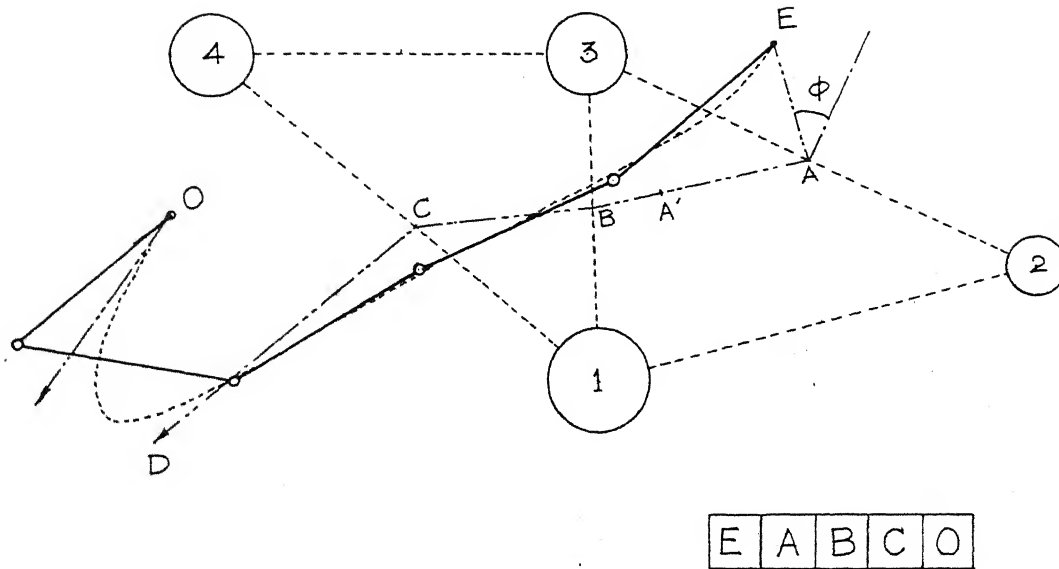


FIG 4.5 : 5-LINK MANIPULATOR

SUPERIMPOSED CONFIGURATIONS



Ideal Points (Q_i)	Joint Positions (M_i)	Bezier Points
204,71	204,71	204,71
171,99	173,97	182.14, 100.06
123,121	137,114	167,03, 105.00
81,139	101,130	131.25, 106.34
62,137	61,136	17.62, 190.03
90,108	90,108	90, 108

FIG 4.6 : BEZIER CURVE AND MANIPULATOR CONFIGURATION FOR CASE IN PROBLEM 1 .

Fig. 4.6 shows the Bezier curve for the Bezier points EA'BCD by the dotted line. Also listed in the same figure are the ideal points Q_1 , the joint positions M_1 and the Bezier points. All the results, except one (with five unequal links), presented in this work have been smoothened as explained above.

Next, a five link manipulator with unequal links is studied. The link lengths are $a_1 = 60$, $a_2 = 50$, $a_3 = 45$, $a_4 = 25$, $a_5 = 20$. The maximum joint rotations have been found to be $35 \leq \theta_1 \leq 172$, $-158 \leq \theta_2 \leq -92$, $-80 \leq \theta_3 \leq 57$, $-9 \leq \theta_4 \leq 34$, $-22 \leq \theta_5 \leq 39$. Fig. 4.7 shows the manipulator configuration at positions 18, 29 and 34 without smoothening the first Bezier point in the list. Fig. 4.8 shows the same manipulator at the same positions, but incorporating the smoothening scheme.

The plots of joint angles for the five links manipulators with end-effector position number are shown in Fig. 4.9. Fig. 4.9a shows the variation of the fifth joint angle with position number. The figure shows plots with unequal links incorporating smoothening, unequal links without smoothening and equal links with smoothening. It is evident that joint rotations without smoothening have very large variations. Fig. 4.9b shows the plots of the fourth joint angles with end-effector position number. The three cases as mentioned earlier have been considered, and it is observed that joint angle for the unequal link case with smoothening has larger variations in this case. This is so because the smaller variation of the fifth joint angle has to be compensated by the fourth joint angle. Fig. 4.9c shows the plots of the

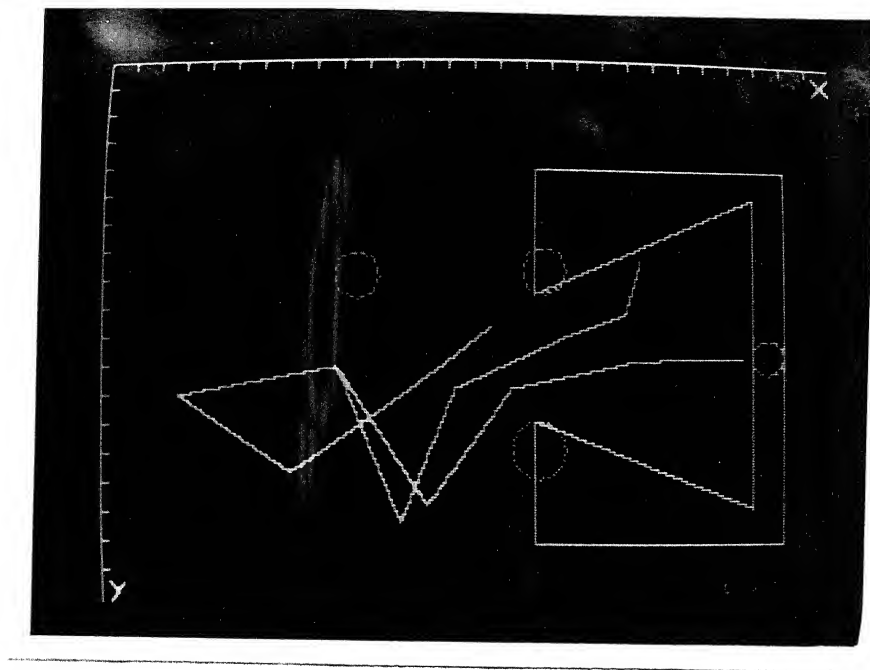


FIG 4.7 : 5 UNEQUAL LINKS
WITHOUT SMOOTHENING

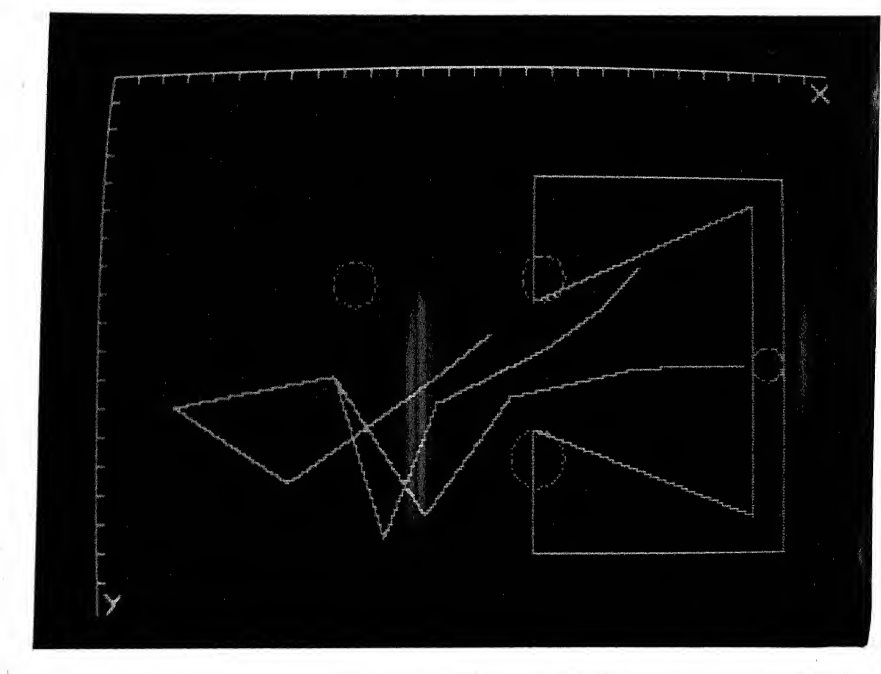


FIG 4.8 : 5 UNEQUAL LINKS
WITH SMOOTHENING

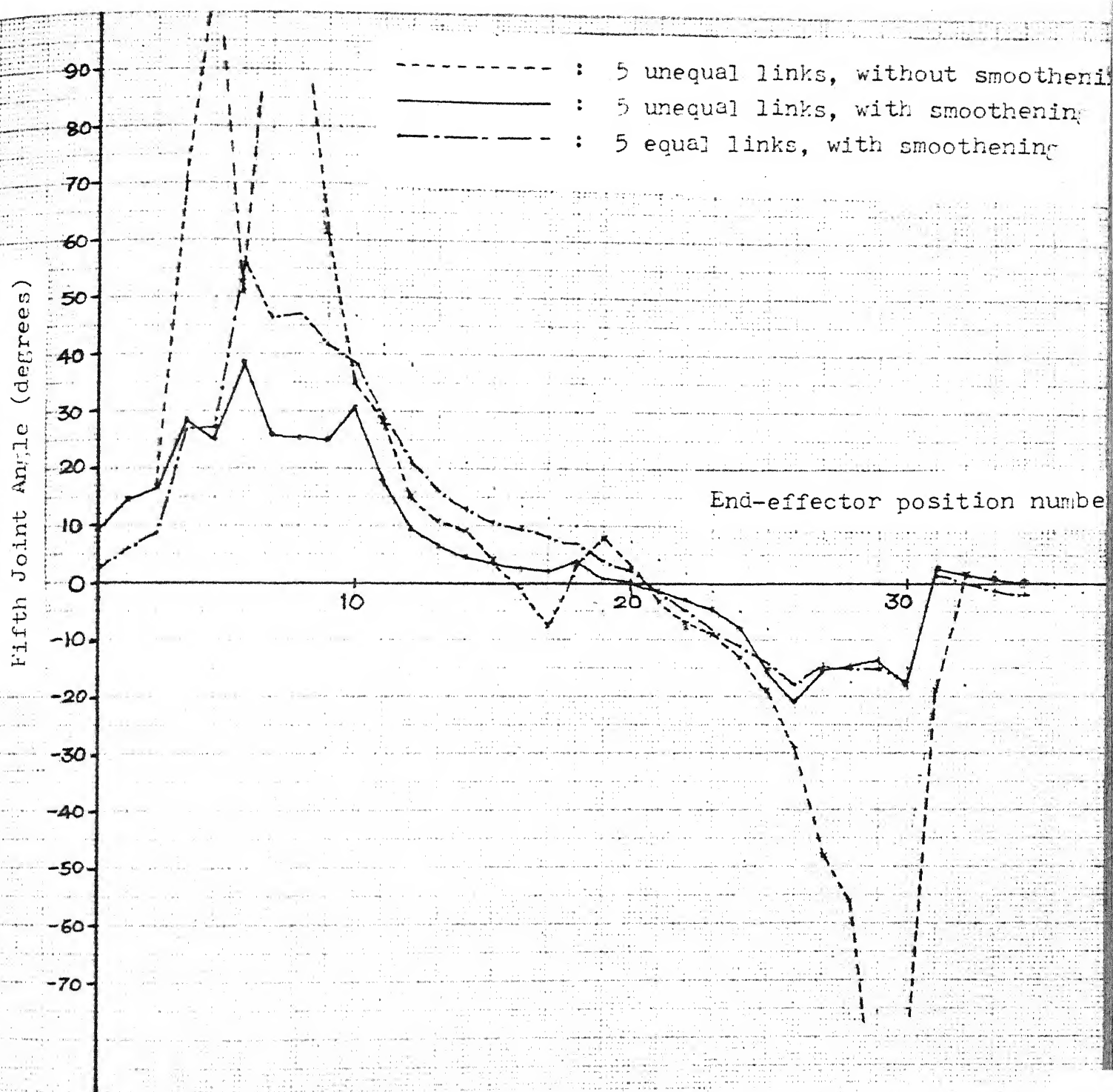


FIG 4.9a : FIFTH JOINT ANGLE vs
EE POSITION NUMBER

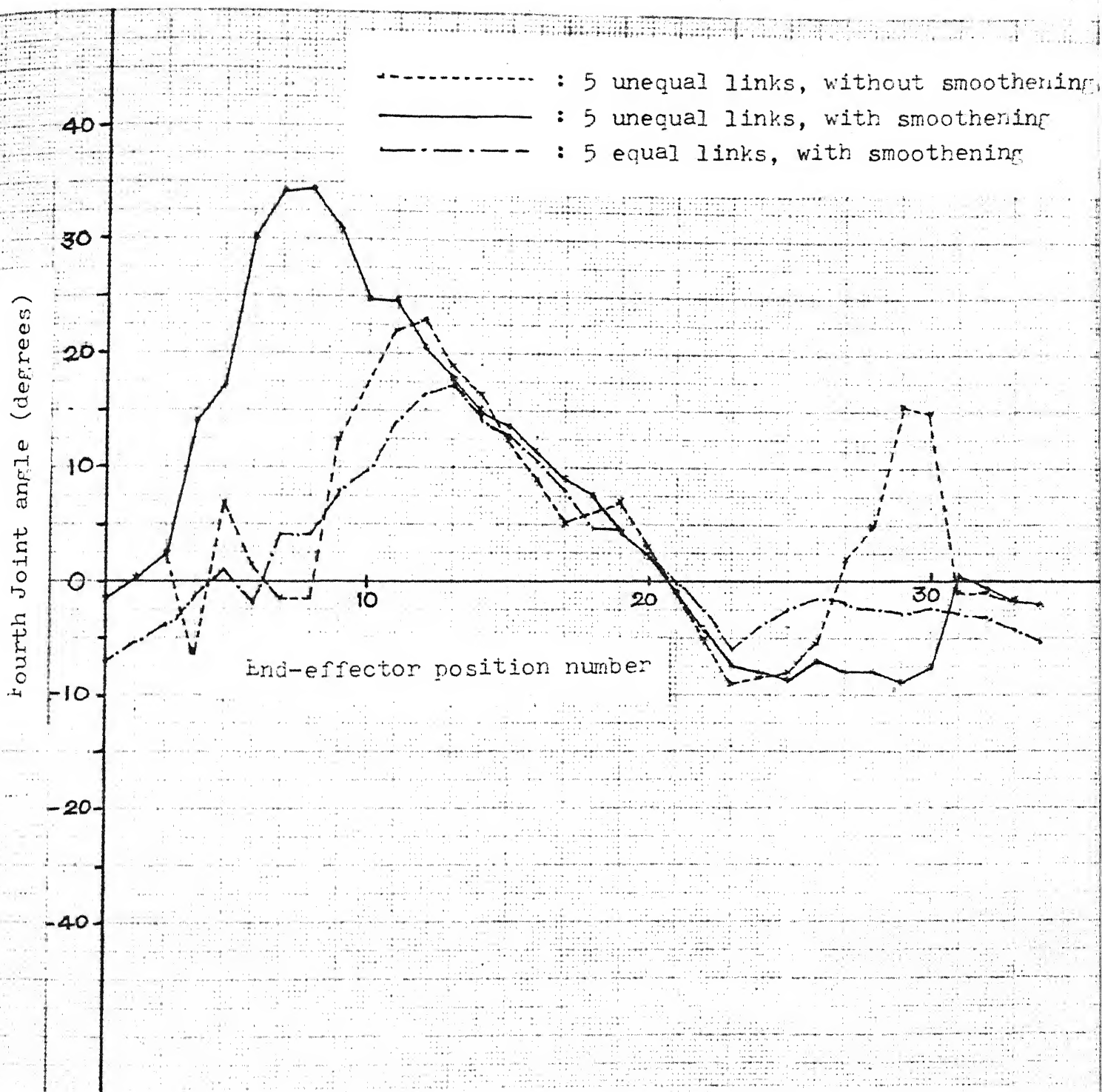


FIG 4.9 b: FOURTH JOINT ANGLE vs
EE POSITION NUMBER

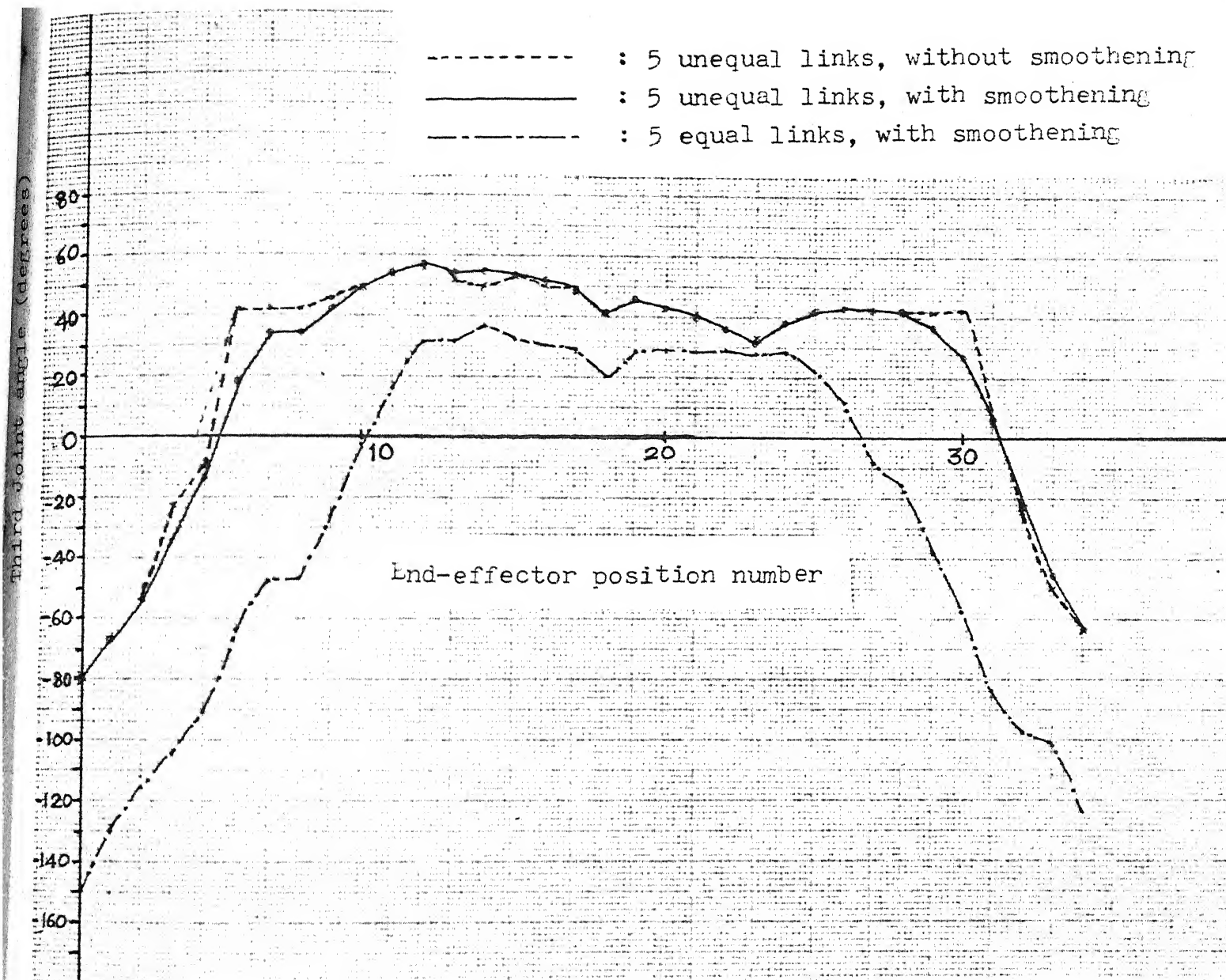


FIG 4.9 c : THIRD JOINT ANGLE vs
EE POSITION NUMBER

third joint angles with end-effector position number. Comparison of the two cases of five unequal link manipulator (i.e., without and with smoothening) reveals that the effect of smoothening is barely reflected at the third joint. It was found that the first and the second joints also retain nearly same motions.

The above problem has also been solved with a 6-link and a 8-link manipulator and the joint angles have been compared. The plots of the last joint angles with end-effector position for the 5, 6 and 8 link manipulators are presented in Fig. 4.10a. It is observed that the last joint angle for all manipulators has the same nature and does not vary significantly. A marginal decrease in joint rotation is evident if the number of links are increased. Fig. 4.10b shows the plot of the $(n-2)^{th}$ joint angle from the end-effector side with end-effector position number, where n is the number of links of the manipulator. It is seen that the variation of this joint angle decreases as the number of links are increased. Four configurations of the six link manipulator during the task, are shown in Fig. 4.11.

One last variation of this problem has been tried with a 5-link manipulator with equal link lengths but bringing the base nearer to the workpiece. The configurations for positions 6, 13 and 27 are given in Fig. 4.12. As expected, the joints need very large rotations and the manipulator configurations look impractical. No attempt is made in this work to place the base of the manipulator at an ideal position. A criterion to

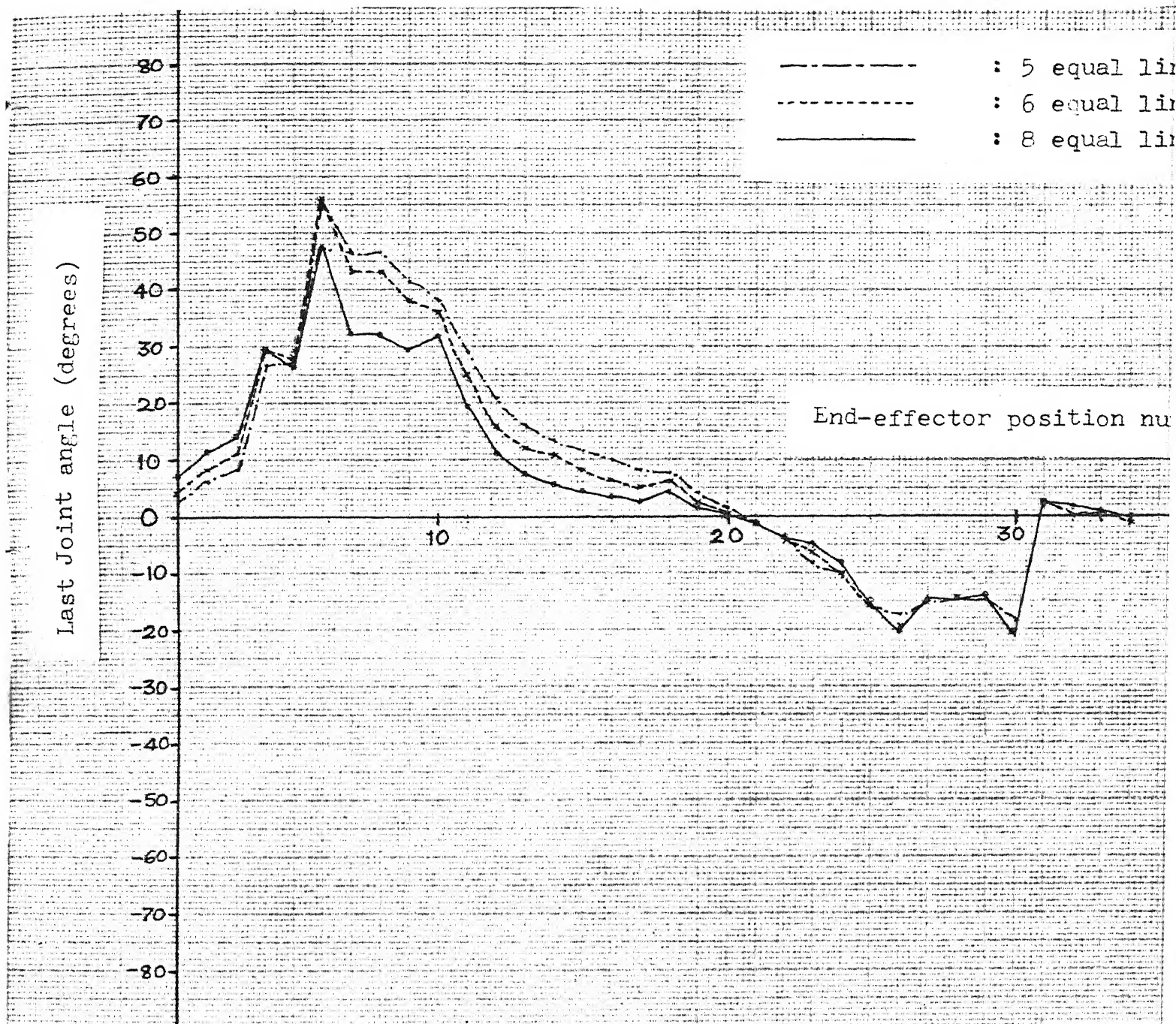


FIG 4.10a : LAST JOINT ANGLE vs
EE POSITION NUMBER

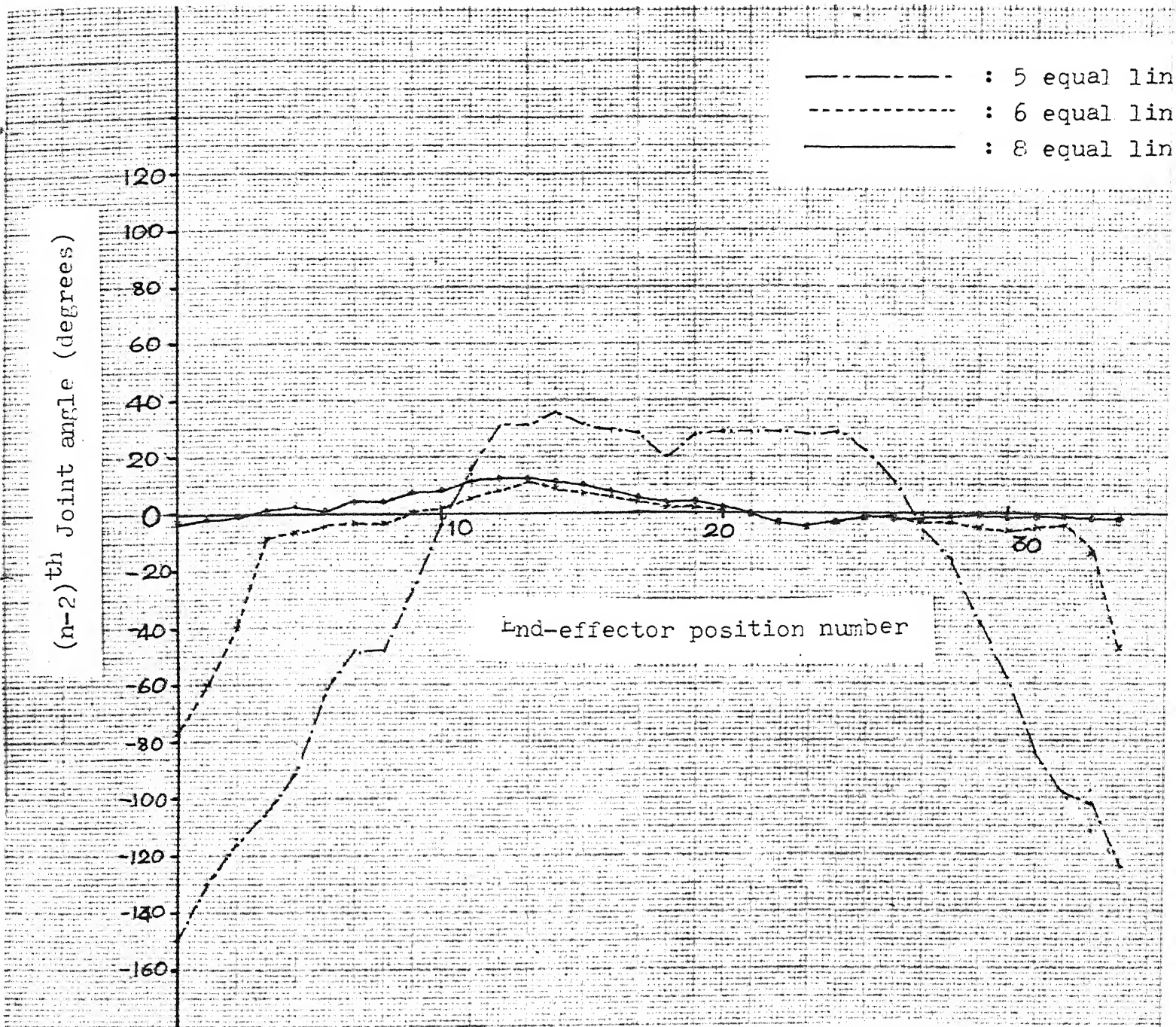


FIG 4.10 b : $(n-2)^{\text{th}}$ JOINT ANGLE vs
EE POSITION NUMBER

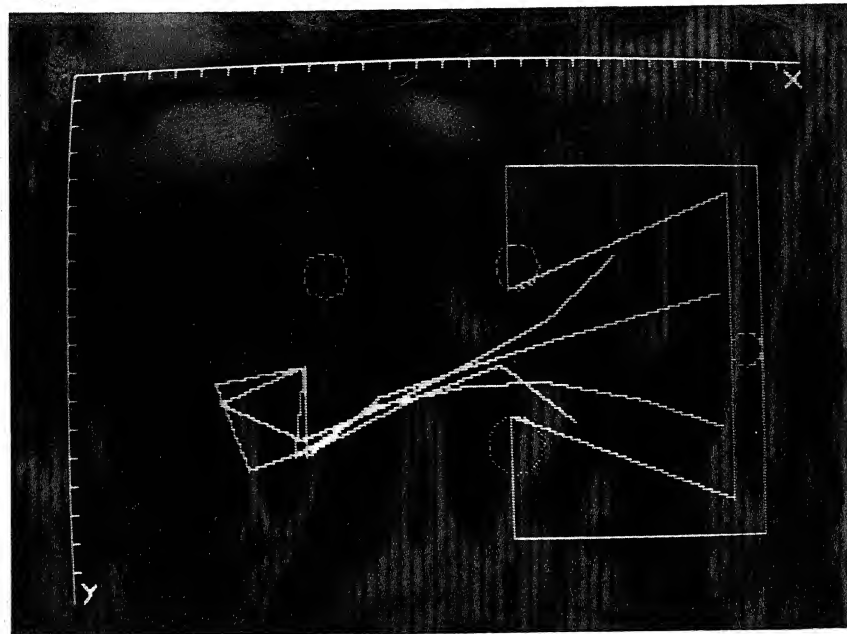


FIG 4.11 : 6-LINK MANIPULATOR
EQUAL LINK LENGTHS

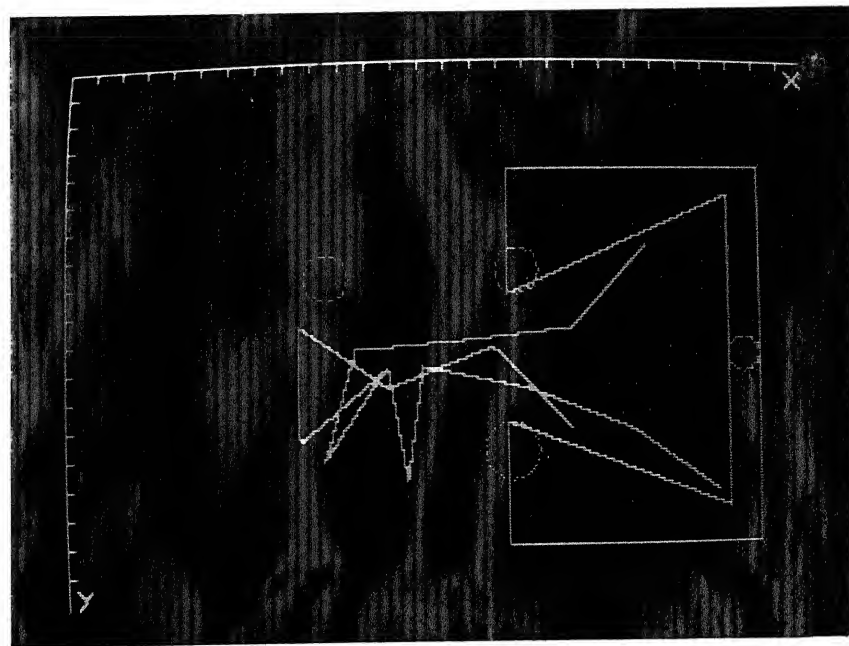


FIG 4.12 : 5-LINK MANIPULATOR , EQUAL ,
BASE NEARER , POSITIONS 6 13 & 27

determine an optimal number and length of links (keeping the total length same) is also not studied here.

The average computation time for the 5, 6 and 8 link manipulator solutions on an Apple IIe micro-computer are given below.

5 - link manipulator	55.5 sec.
6 - link manipulator	64.2 sec.
8 - link manipulator	73.3 sec.

It is seen that the computation time does not increase proportionately with the number of links. The increase in computation time is because after each configuration is determined, the checks are made to ensure collision-free configuration. Each link is checked with respect to each obstacle. A modification can be incorporated to decrease this additional computation time with increase in number of links.

4.3 OBJECT WITH FIVE INTERIOR EDGES:

This object is shown in Fig. 4.13. The workspace model is shown in Fig. 4.14 and is similar to that of Fig. 4.2. Now the operational paths passing through points E_3 and E_6 increase the complexity of the problem. The problem is solved by using a 5-link manipulator. Fig. 4.15 shows the configurations at position 2, 8 and 13 while Fig. 4.16 shows the configurations at positions 19, 24, 31. The manipulator configuration is determined at fortyone equally spaced end-effector positions.

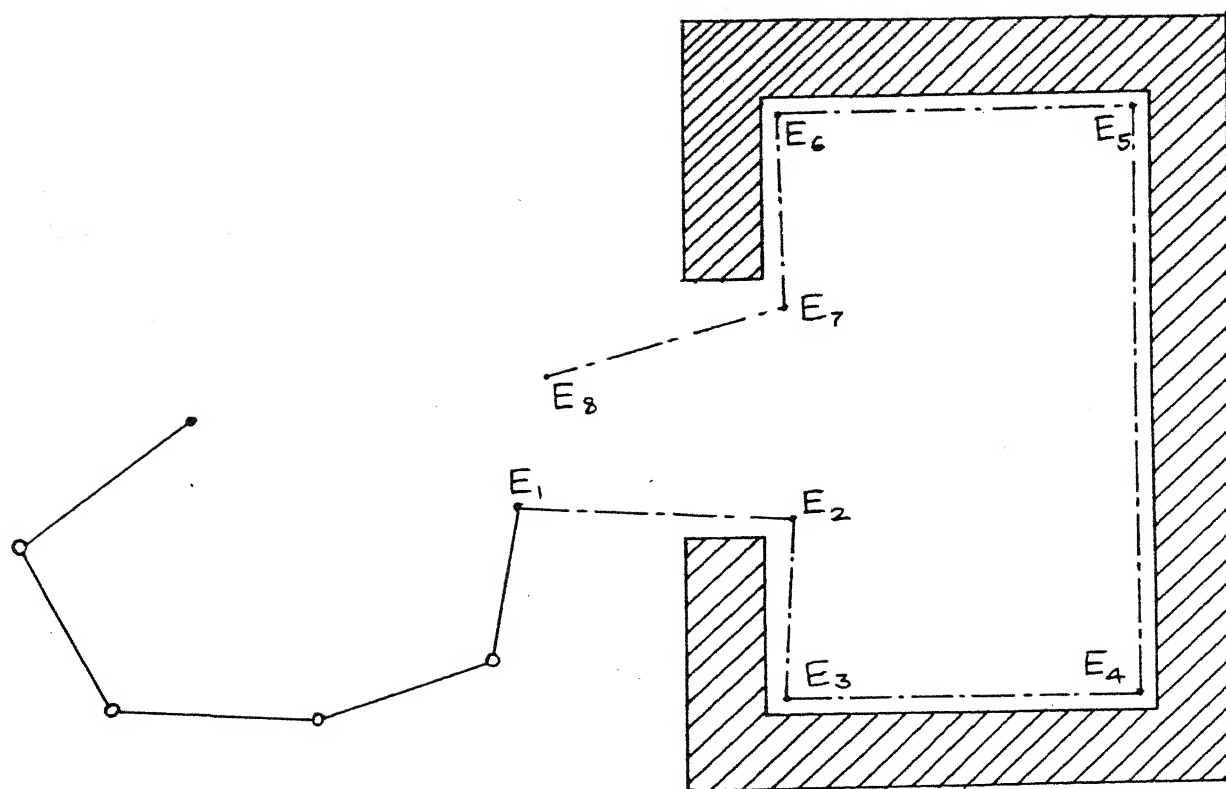


FIG 4.13 : PROBLEM 2 : DEFINITION

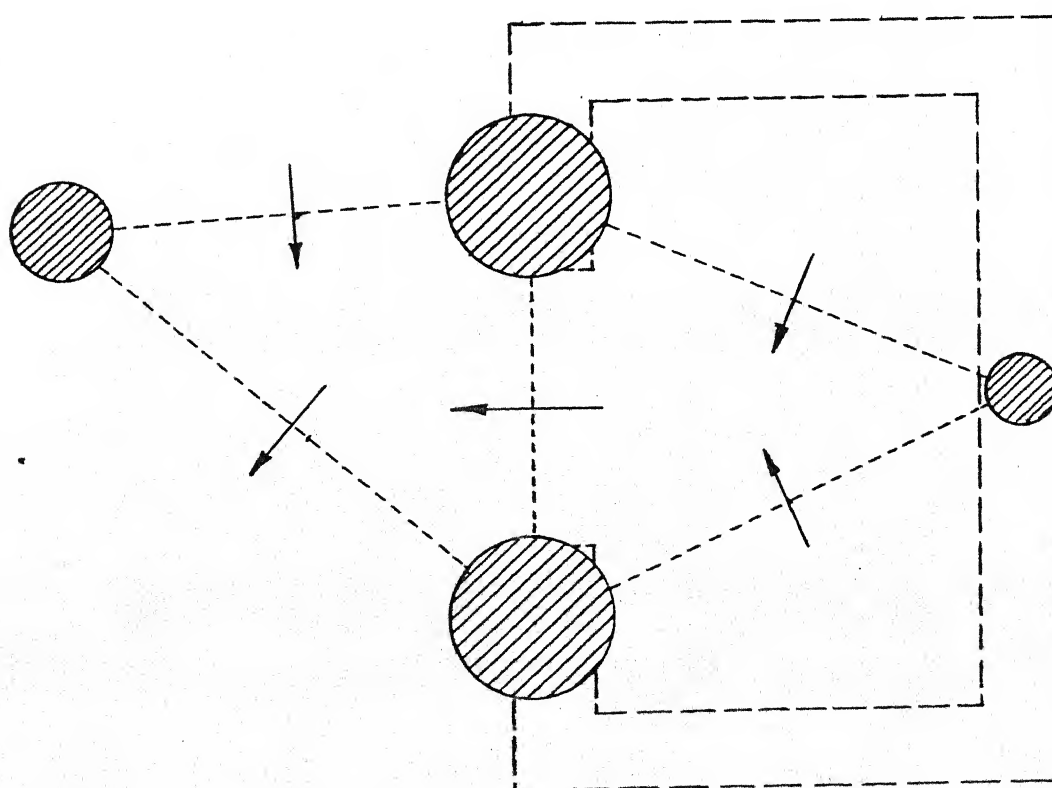


FIG 4.14 : MODELLING OF WORKSPACE

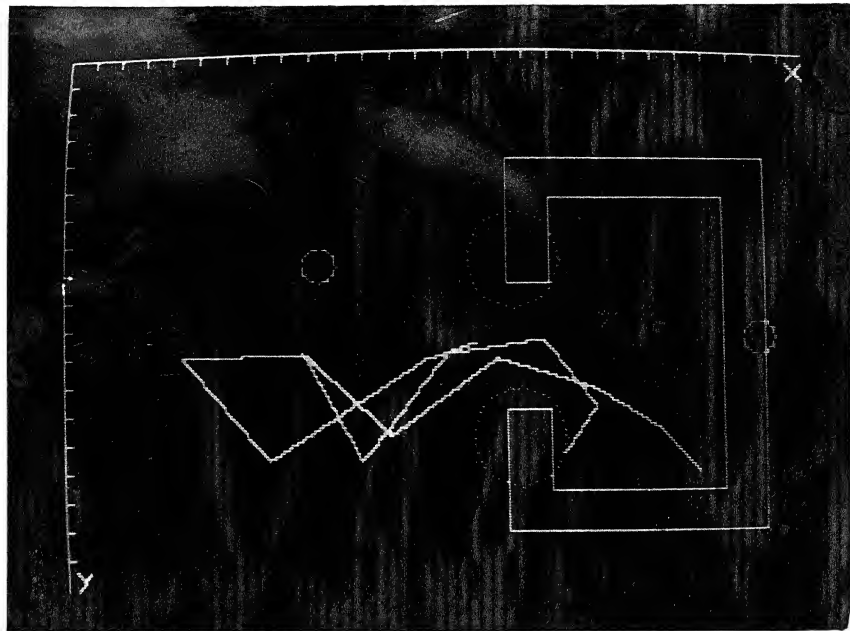


FIG 4.15 : 5-LINK MANIPULATOR,
POSITIONS 2, 8 & 13

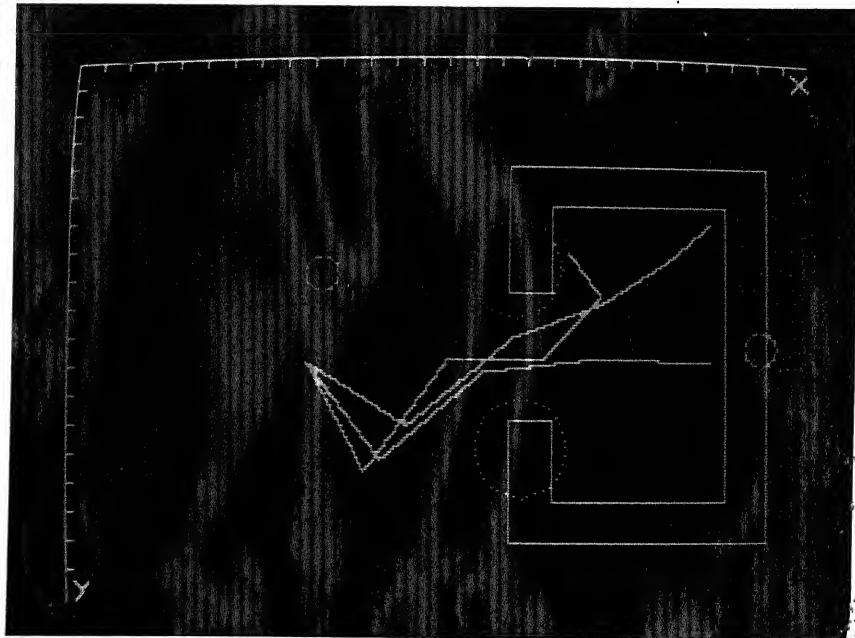


FIG 4.16 : 5-LINK MANIPULATOR,
POSITIONS 19, 24 & 31

At position number 31, the end-effector position is E_a as shown in Fig. 4.17a. At this position, E_a is not visible from O and so a marker needs to be put at the previous edge. In this case the marker has been put at the edge 23 as shown in Fig. 4.17b. Another strategy for putting markers has been adopted here where necessary. At position E_b (Fig. 4.17a), the point is visible from O and so a marker has not been used anywhere. But link five of the resulting configuration collides with obstacle 1 as shown by the firm line in Fig. 4.17a. In such a case a marker has been put at that edge which contains obstacle 1 at its end, and whose mid-point is nearest to obstacle 1. The edge in this case is 13, and so a marker has been put at B as shown in Fig. 4.17c. The resulting configuration, for end-effector at E_b , is shown with a dashed line in Fig. 4.17a and is free of any collision.

4.4 OBJECT REQUIRING NON-OPERATIONAL PATH PLANNING:

The third problem is a spray-painting problem which may occur during a car painting operation. Fig. 4.18 shows the side panel of a car which has an obstacle between the two windows. It is required to paint the entire roof panel of the car passing through the windows. Projected in 2-D, the problem would appear as shown in Fig. 4.19. The windows are open spaces through which the robot arm is free to go inside the car. The task is defined by specifying the end-effector points E_1 to E_7 . These points thus define the "operational path". The modelling of the workspace by placing obstacles is shown in Fig. 4.20.

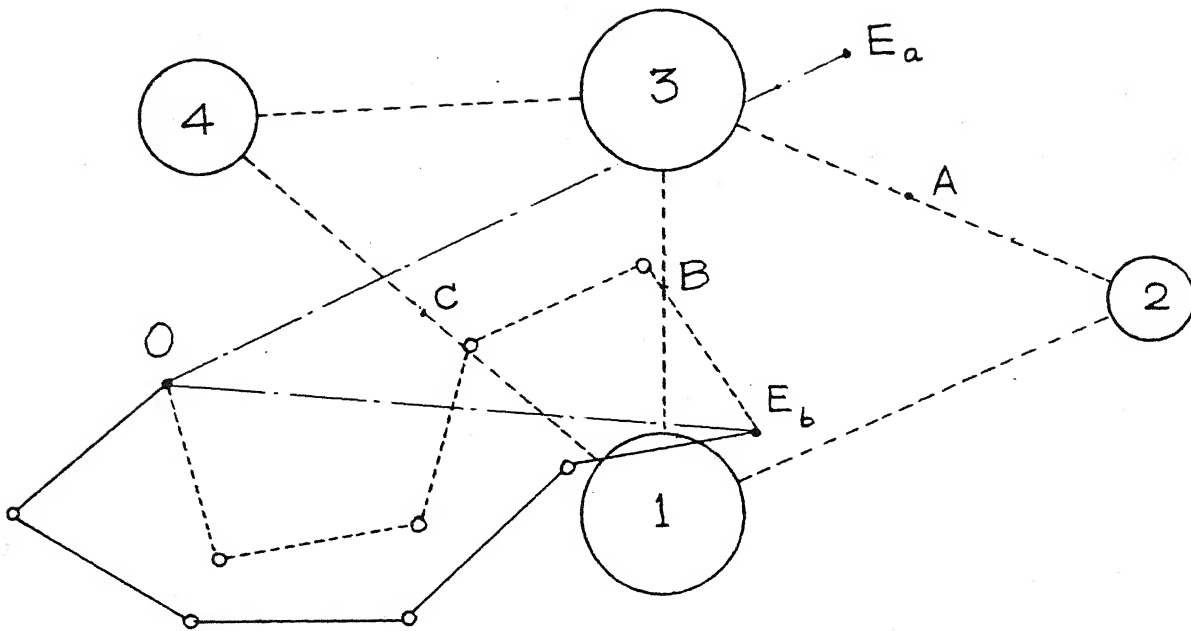


FIG 4.17a DEMONSTRATING MARKERS

	23	13	14	
E_a	A	B	C	O
	✓			

	13	14	
E_b	B	C	O
	✓		

FIG 4.17b : LIST FOR E_a FIG 4.17c : LIST FOR E_b

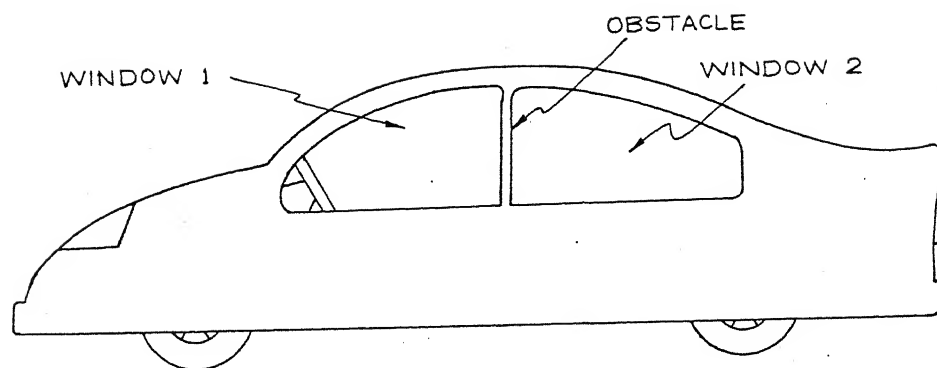


FIG 4.18: PHYSICAL DESCRIPTION OF TASK :PROB.3

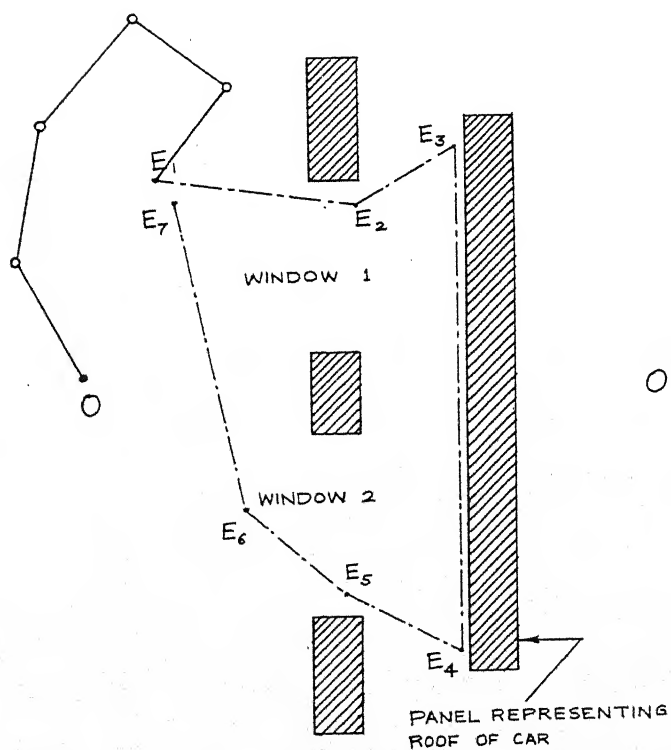


FIG 4.19: PROBLEM DEFINITION
IN 2-D

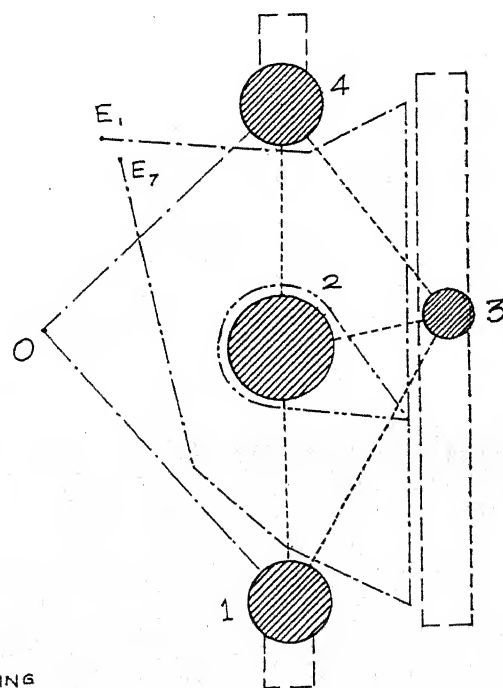


FIG 4.20 MODELLING OF
WORKSPACE

This problem has been solved with a 5-link manipulator and the results are presented in Fig. 4.21 through Fig. 4.24. Fig. 4.21 shows positions 6, 12 and 18 of the manipulator. At position 18, the manipulator cannot access the next end-effector position without colliding with obstacle 2 if it goes in through window 1 (See Fig. 4.19 and 4.20). The non-operational path is now calculated to circumvent the obstacle 2, and its rough nature is shown in Fig. 4.20 by double chain dotted line. The actual operational and non-operational paths, as calculated, are shown in frame of Fig. 4.22. The total number of end effector positions on the operational and non-operational path is 57 and these points are equally spaced. Fig. 4.23 shows position 20, 25 and 27 which lie on the non-operational path. Fig. 4.24 shows three positions on the non-operational path when it is approaching the panel through window 2. Fig. 4.25 shows positions 35, 39 and 45 on the operational path when the manipulator is retracting to its starting position. Configurations at various stages of the task have been collectively shown in Fig. 4.26.

A major difference in the three problems presented is that in the first two cases, the entire end-effector trajectory lies inside the outer polygon. However, the obstacles for the car-roof painting problem are placed in such a way that there are positions of the end-effector which lie outside the outer polygon but are still in zones 2 to 4. In such cases, if the algorithm of Case 4 (Sec. 2.4.2) is followed, the manipulator may collide

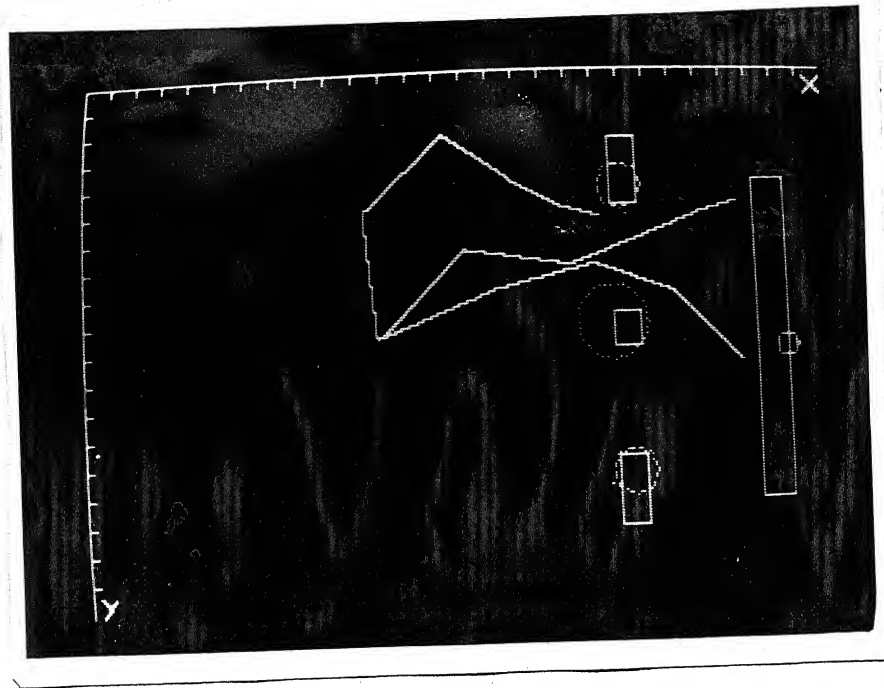


FIG 4.21: 5-LINK MANIPULATOR
POSITIONS 6, 12 & 18

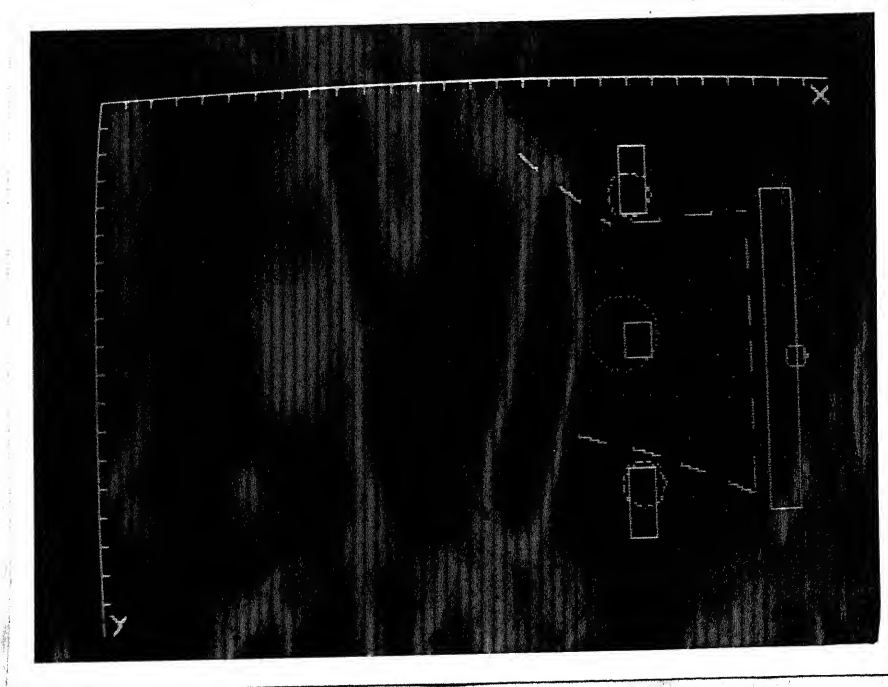


FIG 4.22 : OPERATIONAL AND NON-OPERATIONAL
PATH

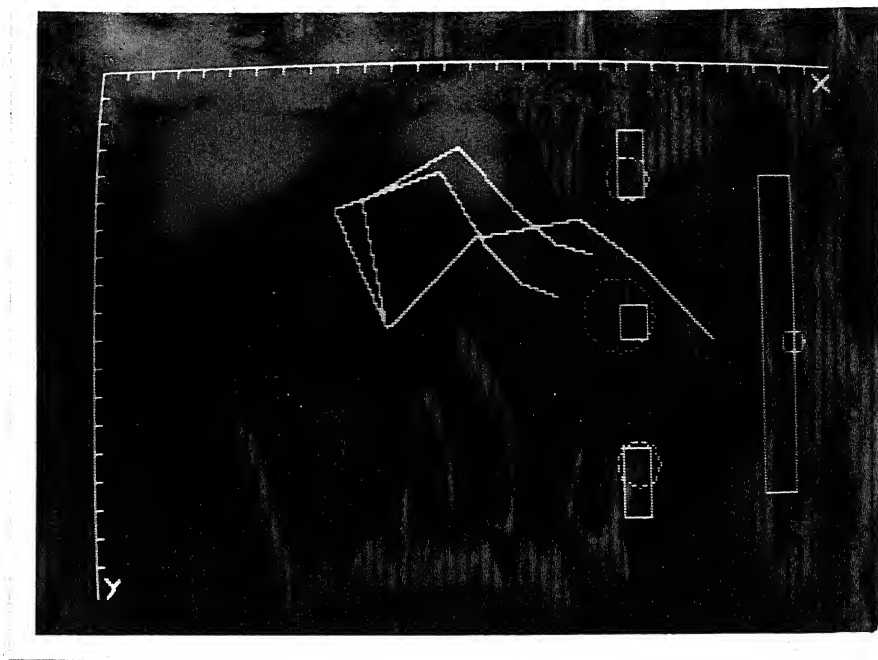


FIG 4.23: 5-LINK MANIPULATOR
POSITIONS 20, 25 & 27

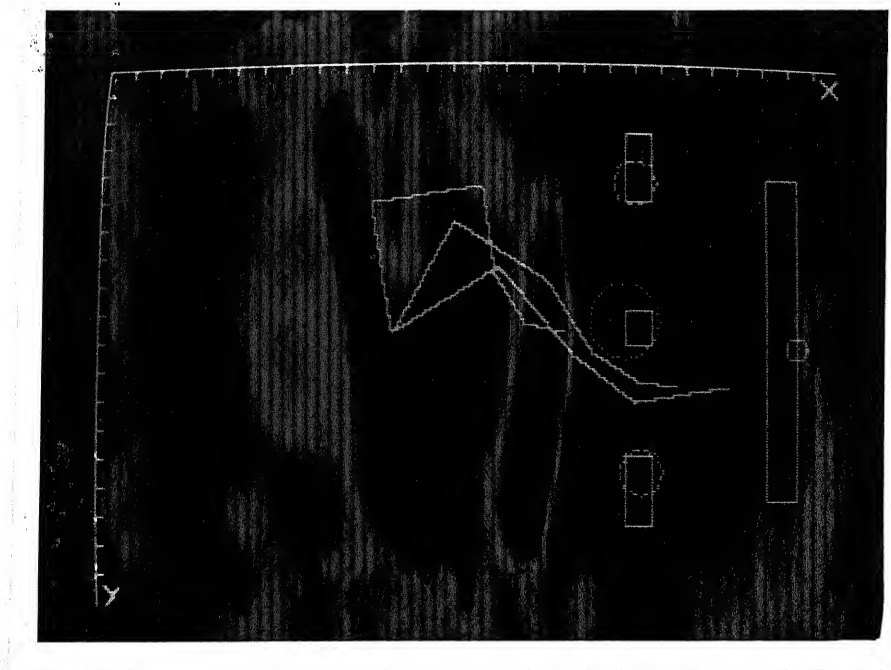


FIG 4.24: 5-LINK MANIPULATOR
POSITIONS 28, 33, 34

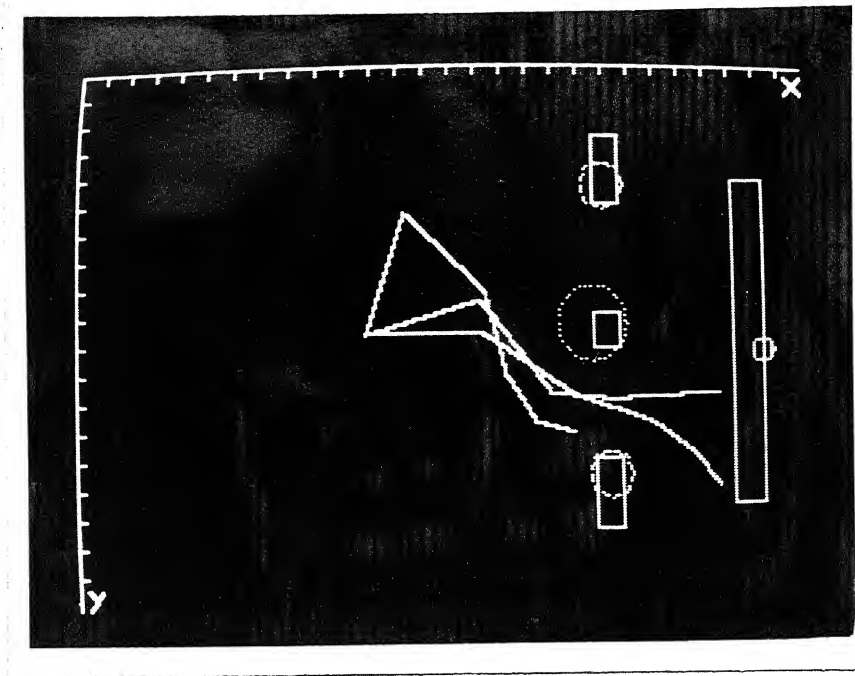


FIG 4.25 : 5-LINK MANIPULATOR
POSITIONS 35,39 & 45

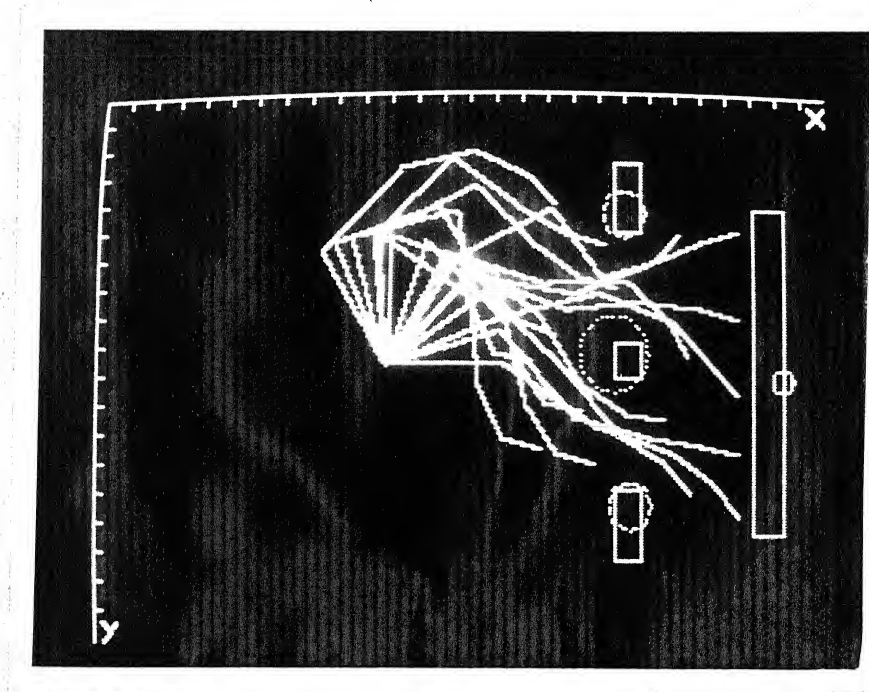


FIG 4.26 5-LINK MANIPULATOR
SUPERIMPOSED CONFIGURATIONS

with an obstacle. To avoid this possibility altogether, a Bezier point is placed on the arrow corresponding to the edge nearest to base 0 in the list. The distance of this Bezier point from the centre of arrow is taken as two and half times the length of the arrow.

CHAPTER V

CONCLUSIONS

A method is proposed in this work to solve the inverse problem of determining joint angles of a manipulator with redundant links for a given end-effector position. Redundant links are used to circumvent obstacles in the workspace. Three different examples in increasing order of difficulty have been successfully solved using the proposed algorithm. For each example, the end-effector is required to trace surfaces inside some objects. The end-effector path is approximated by sufficiently close points and the safe manipulator configuration is determined at each point. Further, an example has been illustrated to identify a case where the end-effector cannot access the next point from the present configuration without the links colliding with some obstacle. Such cases have also been taken care of by planning a non-operational path for the end-effector. The algorithm determines successive manipulator positions for this non-operational path during which the manipulator retracts and approaches the same end-effector position after circumventing obstacles. In author's knowledge, this type of case has not been dealt with in earlier studies.

Unlike previously reported literature, the proposed algorithm is not much dependent on the number of links in the manipulator, thus making it most suitable if the number of redundant links is large. Given obstacles in the workspace and the total length of links, the major part of the algorithm

determines a suitable smooth curve avoiding all the obstacles. This curve is independent of number of links. Approximating this curve by the links is easy and care has been taken to recheck for collision.

It was observed that by increasing the number of links, while keeping the total length constant, the joint rotations of the links towards end-effector can be reduced. The links towards base remain mostly unaffected. Because greater manoeuvrability is required on the end-effector side, the increase in number of links distributes the motion towards end of the manipulator.

Finally, the proposed method is expected to always yield a safe configuration of the manipulator, if such a configuration is feasible.

Scope of Future Work:

Instead of modelling the objects with discrete circular obstacles, concept of barrier edges should be incorporated which ensures that arrows at that edge always point toward the free space enclosing the end-effector.

Given the total length of the links, determination of the optimum number of links and optimum link lengths would help in identifying the ideal manipulator suited for a task.

Lastly, for each object or task, there may be an optimum position of the base of manipulator so that the task is performed with minimum joint rotations.

BIBLIOGRAPHY

1. Loeff, L.A., and Soni, A.H., An algorithm for computer guidance of a manipulator in between obstacles, *Journal of Engineering for Industry, Series B*, 97(3), pp.836-842, 1975.
2. Khatib, O., Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, *IEEE International Conference on Robotics and Automation*, pp. 500-505, 1985.
3. Kircanski, M. and Vukobratovic, M., Trajectory Planning for Redundant Manipulators in the Presence of Obstacles, *Theory and Practice of Robots and Manipulators, Proc. of RoManSy'84: The Fifth CISM-IFT0 MM Symposium*, pp. 57-63.
4. Maciejewski, Anthony A. and Klein, Charles A., Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments, *The International Journal of Robotics Research*, Vol. 4, No. 3, pp. 109-117, Fall 1985.
5. Brooks, Rodney A. and Lozano-Perez, Tomas, A Subdivision Algorithm in Configuration Space for Findpath with Rotation, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-15, No. 2, pp. 224-233, March/April 1985.
6. Brooks, Rodney A., Solving the Find-Path Problem by Good Representation of Free Space, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-13, No. 3, pp. 190-196, March/April, 1983.
7. Giralt, George, *Mobile Robots, Robotics and Artificial Intelligence*, NATO ASI Series, Springer-Verlag Berlin Heidelberg, New York, Tokyo, pp. 363-393, 1984.
8. Pachner, Jaroslav, *Handbook of Numerical Analysis Applications with Programs for Engineers and Scientists*, McGraw Hill Co., ISBN-0-07-048057-5, 1984.

APPENDIX I

BEZIER CURVES

A Bezier curve of order m is represented by the expression,

$$\vec{r}(u) = \sum_{i=0}^m \vec{r}_i {}^m C_i (1-u)^{m-i} u^i \quad (A1.1)$$

where u is a parameter that varies along the length of the curve, ($0 \leq u \leq 1$). \vec{r}_i is the vector coordinates of the i^{th} input point, and $\vec{r}(u)$ represents the vector coordinates of a point on the Bezier curve.

A curve of order m requires $(m+1)$ input points which may be called Bezier Points. The Bezier curve need not necessarily pass through the Bezier points. Fig. A1.1 shows a planar Bezier curve of order 3. The parameter u is zero at \vec{r}_0 and equal to 1 at \vec{r}_3 . At all other intermediate points $0 < u < 1$,

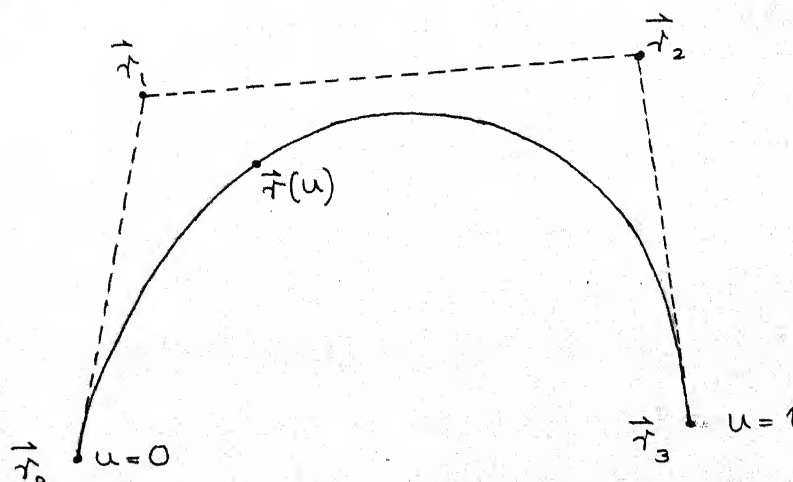


Fig. A1.1: Bezier Curve of Order 3.

separate file with the suffix ".PIK".

The commands to input the above information is listed below:

A	to	enter the base coordinates
Z	to	enter end-effector locations. The actual path is obtained by joining them in sequence.
M	to	enter manipulator joint position. The number of links is incremented by one and link length is decided by joining present point to the last joint position.
C	to	cancel previous Z or M command if immediately followed by it.
←	to	move cursor to the left
→	to	move cursor to the right
↑	to	move cursor up
↓	to	move cursor down
S	to	decrease speed of cursor movement
F	to	increase speed of cursor movement
O	to	input obstacle position. This will enable only the ↑, ↓ and R keys.
R	to	decide radius and draw a circle at present obstacle location
P	to	plot a point at present cursor position
D	to	draw a point from previous plotted (or drawn) to present cursor position
X	to	cancel previous point plotted or drawn
ESC	to	exit from graphical input mode.

The program will ask for a file name where the data is to be stored. The input data will be stored in a filename which will have the suffix ".DAT" while the picture is stored with the suffix ".PIC".

The program then begins forming the network and asks the user for the starting viewing angle which has been defined in Sec. 2.2.4. Any number from 0 to 360 may be entered. The list forming the convex polygon is displayed, and after a while the program informs the user that the network is ready.

While giving the obstacle locations, please take care that the slope of the line joining any two obstacles is not zero or infinity to avoid an overflow error.

The network is next displayed and the arrows are shown with a dot at the tail first. On pressing a key the rest of the arrow is drawn. The direction of the arrow can thus be ascertained.

If the network is satisfactory, it is automatically stored on disk with the suffix ".NET" when the Y key is pressed. Otherwise the program can be run again and the "starting viewing angle" can be changed to yield a new network. The second time, input data may be directly read from disk and need not be entered manually.

After forming the network, the robot configuration can be solved by typing.

] RUN ROUTINE)

This program will ask for the file name where the network has been stored. It reads the stored networks, and asks for the limits on joint rotations which have to be entered from the keyboard. The limits are entered in degrees. If the limit for any joint is entered as 50 (for instance), then the limit will be taken as +50 to -50 degrees.

The program continues execution now, informing at each stage the status of the end-effector. A detailed listing of all parameters appear on the screen when the "end-effector natural state number" changes. The beginning and end of a state change operation is also informed with a hoot. For fast execution, there is a provision for jumping to the next input point on the end-effector path while it is tracking the "operational path". This may be done by simply pressing any key during execution. The coordinates of joints are automatically stored on disk with the suffix ".JNT".

The configurations at various end-effector positions or a plot of all joint angles w.r.t. time can be viewed through a program called PLOTTER. To invoke this program type

] RUN PLOTTER)

The filename has to be entered first and then the program asks for a choice of display mode. The menu is self-explanatory. In choice no. 4, if all the angles wish to be plotted at the same time on the screen, any number which is beyond the total number of joints may be entered. The step length will depend on the total number of end-effector positions that have been calculated.

APPENDIX III

```

1  REM *****
2  REM *
3  REM * OBSTACLE AVOIDANCE *
4  REM * PROGRAM *
5  REM *
6  REM * Anupam Bagchi *
7  REM * Dec 1986 *
8  REM *
9  REM *****
10 DIM XE(15),YE(15),XB(10),YB(10),RSZ(10),XIZ(20),YIZ(20),AA(10)
15 PI = 3.14159265:BB = 180 / PI
20 REM Cubic Bezier Curve
30 TEXT : NORMAL : HOME
40 VTAB 4:A$ = "CHOOSE INPUT METHOD": GOSUB 9000: VTAB 8: HTAB 5: PRINT
   "Press": VTAB 10: HTAB 8: PRINT "G to give input graphically": PRINT
   : HTAB 8: PRINT "D to read from disk": PRINT : HTAB 8: PRINT "K
   to type from keyboard ";
45 GET KE$: IF KE$ = "G" OR KE$ = "D" OR KE$ = "K" THEN 50
47 GOTO 45
50 IF KE$ = "G" THEN GOSUB 3000: GOTO 150
55 IF KE$ = "K" THEN GOSUB 1200: GOTO 150
60 GOSUB 9500: GOSUB 1100: GOTO 160
150 GOSUB 1100: GOSUB 8000: GOSUB 8200
160 GOSUB 41000: GOSUB 10000: GOSUB 10030: GOSUB 10550
165 REM GOSUB 50200: FOR SS = 2 TO P3%: HPL0T XB(S3%(SS - 1)),YB(S3%(SS
   - 1)) TO XB(S3%(SS)),YB(S3%(SS)): NEXT SS: HPL0T XB(S3%(P3%)),YB(S3%
   (P3%)) TO XB(S3%(1)),YB(S3%(1))
170 PRINT : PRINT : HTAB 7: PRINT "List forming convex polygon": PRINT :
   HTAB 7: PRINT "Obstacles ";; FOR JJ = 1 TO P3%: HTAB 17: PRINT S3%(
   JJ): NEXT JJ: HTAB 17: PRINT "...in sequence."
180 GOSUB 10700: FOR HH = 1 TO TR%:WT$ = TR$(HH): GOSUB 11000: NEXT HH
185 A$ = "Network is ready !": PRINT CHR$(7): GOSUB 9000: PRINT : PRINT
   : HTAB 7: PRINT "Hit a key to plot network ";; GET KE$
190 GOSUB 50200
200 FOR KK = 1 TO TR%:V1% = VAL ( LEFT$ (TR$(KK),1)):V2% = VAL ( MID$
   (TR$(KK),2,1)):V3% = VAL ( RIGHT$ (TR$(KK),1)): HPL0T XB(V1%),YB(V1
   %) TO XB(V2%),YB(V2%): HPL0T XB(V2%),YB(V2%) TO XB(V3%),YB(V3%)
210 HPL0T XB(V3%),YB(V3%) TO XB(V1%),YB(V1%): NEXT KK: GET KE$
220 FOR KK = 1 TO ED%: HPL0T TX(KK),TY(KK): GET KE$: HPL0T TX(KK),TY(KK)
   TO HX(KK),HY(KK): NEXT KK
230 GET KE$: TEXT :A$ = "Are you satisfied with the results ?": GOSUB 90
   00
240 HTAB 15: PRINT "(Yes/No) ";
250 GET KE$: IF KE$ = "Y" OR KE$ = "N" THEN 265
260 GOTO 250
265 IF KE$ = "Y" THEN 900
270 PRINT : PRINT :A$ = "You may start again by changing": GOSUB 9000:A$
   = "the initial scanning angle.": GOSUB 9000: PRINT : PRINT "YOUR DA
   TA IS STORED IN FILE : ";FI$
280 PRINT : PRINT : HTAB 5: PRINT "Hit a key to restart execution ";; GET
   KE$: RUN
900 GOSUB 2000
1000 END
1100 TEXT : HOME : VTAB 5: HTAB 5: PRINT "Starting point : (";XD$;",";YO$
   ")": PRINT : HTAB 7: PRINT "Number of end points = ";EC%: PRINT : FOR
   JJ = 1 TO EC%: HTAB 7: PRINT "End point ";JJ;" at (";XE(JJ);",";YE(J
   J);")": NEXT JJ
1105 PRINT : HTAB 8: PRINT "Press RETURN for more ";; GET KE$: PRINT
1110 PRINT : HTAB 7: PRINT "Number of obstacles = ";OC%: PRINT : FOR JJ =
   1 TO OC%: HTAB 2: PRINT "Obstacle ";JJ;" at (";XB(JJ);",";YB(JJ);")
   with radius ";RSZ(JJ): NEXT JJ
1115 PRINT : HTAB 8: PRINT "Press RETURN for more ";; GET KE$: PRINT

```



```

1150 PRINT : HTAB 6: PRINT "Press RETURN to continue "; GET KE$: RETURN
1200 REM accepts input from keyboard
1210 HOME : VTAB 5: HTAB 6: PRINT "Base coordinates :": PRINT : HTAB 10:
INPUT "X = ";XO: HTAB 10: INPUT "Y = ";YO
1220 PRINT : HTAB 3: INPUT "Number of end effector positions = ";ECZ: FOR
KK = 1 TO ECZ: PRINT : HTAB 8: PRINT "X end [";KK;" ] = ";: INPUT "":
XE(KK): HTAB 8: PRINT "Y end [";KK;" ] = ";: INPUT "": YE(KK): NEXT KK
: PRINT
1230 PRINT : HTAB 3: INPUT "Number of obstacles = ";OCZ: FOR KK = 1 TO O
CZ: PRINT : HTAB 8: PRINT "X obstacle [";KK;" ] = ";: INPUT "": XB(KK)
: HTAB 8: PRINT "Y obstacle [";KK;" ] = ";: INPUT "": YB(KK): HTAB 12:
PRINT "Radius [";KK;" ] = ";: INPUT "": RSZ(KK): NEXT KK: PRINT
1240 PRINT : HTAB 3: INPUT "Number of links in manipulator = ";NLZ: FOR
KK = 1 TO NLZ: PRINT : HTAB 8: PRINT "Length of link ";KK;" = ";: INPUT
"": AA(KK): NEXT KK: PRINT
1280 RETURN
2000 REM store the network on disk
2010 DO$ = CHR$(4):FO$ = FI$ + ".NET": PRINT DO$: PRINT DO$;"OPEN":FO$:
PRINT DO$;"WRITE":FO$
2020 PRINT XO;"",YO;"",ECZ: FOR JJ = 1 TO ECZ: PRINT XE(JJ);"",YE(JJ)
: NEXT JJ: PRINT OCZ: FOR JJ = 1 TO OCZ: PRINT XB(JJ);"",YB(JJ);"",
RSZ(JJ);"",RD(JJ);"",AN(JJ): NEXT JJ
2030 PRINT NLZ: FOR JJ = 1 TO NLZ: PRINT AA(JJ): NEXT JJ: PRINT EDZ: FOR
JJ = 1 TO EDZ: PRINT ED$(JJ);"",TMZ(JJ);"",TNZ(JJ);"",HX(JJ);"",
HY(JJ);"",TX(JJ);"",TY(JJ);"",EX(JJ);"",EY(JJ): NEXT JJ
2040 PRINT DO$;"CLOSE":FO$
2050 RETURN
2760 ONERR GOTO 2900
2770 BB = 57.2957796: FOR JJ = 0 TO 360 STEP 12:XP = XJ + RJZ * COS (JJ /
BB):YP = YJ + RJZ * SIN (JJ / BB): HPLLOT XJ,YJ TO XP,YP: NEXT JJ: RETUR

2900 REM error handling routine
2910 IF XJ > 279 THEN XJ = 279: RESUME
2920 IF XJ < 0 THEN XJ = 0: RESUME
2930 IF YJ > 191 THEN YJ = 191: RESUME
2940 IF YJ < 0 THEN YJ = 0
2950 RESUME
3000 HOME :ICZ = 10:GLZ = .1:OCZ = 0:ECZ = 0:XO = 0:YO = 0:XM = XO:YM = Y
O:NLZ = 0:KIZ = 0
3010 HGR2 : HCOLOR= 3
3020 BL$ = CHR$(7):XH = 140:YH = 95:XQ = 0:YQ = 0: HPLLOT XH,YH
3030 GOSUB 3150
3040 GOSUB 3980
3045 IF KEZ = 27 THEN RETURN
3050 IF KEZ = 8 THEN GOSUB 3190: GOTO 3040
3060 IF KEZ = 21 THEN GOSUB 3230: GOTO 3040
3065 IF KEZ = 88 THEN 3600
3070 IF KEZ = 10 THEN GOSUB 3270: GOTO 3040
3075 IF KEZ = 77 THEN 3550
3085 IF KEZ = 79 THEN 3410
3090 IF KEZ = 11 THEN GOSUB 3310: GOTO 3040
3095 IF KEZ = 65 AND NLZ = 0 THEN XO = XH:YO = YH:XM = XO:YM = YO:NSZ =
1: GOTO 3040
3110 IF KEZ = 80 OR KEZ = 68 THEN 3350
3115 IF KEZ = 90 THEN 3500
3120 IF KEZ = 70 THEN 3380
3130 IF KEZ = 83 THEN 3390
3140 GOTO 3040
3150 HPLLOT 0,191 TO 0,0 TO 279,0
3160 HPLLOT 279,3 TO 279,10: HPLLOT 279,3 TO 279,10: HPLLOT 3,184 TO 7,188:

```

```

3170 FOR JJ = 10 TO 270 STEP 10: HPLLOT JJ,0 TO JJ,3: NEXT : FOR JJ = 10 TO
180 STEP 10: HPLLOT 0,JJ TO 3,JJ: NEXT
3180 RETURN
3190 IF XH - ICZ < = 4 THEN PRINT BL$;: RETURN
3200 IF NSZ = 1 THEN 3220
3210 HCOLOR= 0: HPLLOT XH,YH: HCOLOR= 3
3220 XH = XH - ICZ: HPLLOT XH,YH: NSZ = 0: RETURN
3230 IF XH + ICZ > = 279 THEN PRINT BL$;: RETURN
3240 IF NSZ = 1 THEN 3260
3250 HCOLOR= 0: HPLLOT XH,YH: HCOLOR= 3
3260 XH = XH + ICZ: HPLLOT XH,YH: NSZ = 0: RETURN
3270 IF YH + ICZ > = 191 THEN PRINT BL$;: RETURN
3280 IF NSZ = 1 THEN 3300
3290 HCOLOR= 0: HPLLOT XH,YH: HCOLOR= 3
3300 YH = YH + ICZ: HPLLOT XH,YH: NSZ = 0: RETURN
3310 IF YH - ICZ < = 4 THEN PRINT BL$;: RETURN
3320 IF NSZ = 1 THEN 3340
3330 HCOLOR= 0: HPLLOT XH,YH: HCOLOR= 3
3340 YH = YH - ICZ: HPLLOT XH,YH: NSZ = 0: RETURN
3350 IF NSZ = 1 THEN 3040
3355 KIZ = KIZ + 1: XIZ(KIZ) = XH: YIZ(KIZ) = YH: IF KEZ = 68 AND KIZ > 1 THEN
HPLLOT XIZ(KIZ - 1), INT ((YIZ(KIZ - 1) / 1000 - INT (YIZ(KIZ - 1) /
1000)) * 1000 + .05) TO XIZ(KIZ), YIZ(KIZ): GOTO 3370
3360 YIZ(KIZ) = YIZ(KIZ) + 1000
3370 NSZ = 1: GOTO 3040
3380 ICZ = ICZ + 1: GOTO 3040
3390 IF ICZ < = 1 THEN 3040
3400 ICZ = ICZ - 1: GOTO 3040
3410 OCZ = OCZ + 1: NSZ = 1: XB(OCZ) = XH: YB(OCZ) = YH
3420 GOSUB 3980: IF KEZ = 67 THEN NSZ = 0: OCZ = OCZ - 1: GOTO 3040
3430 IF KEZ = 10 THEN GOSUB 3270: GOTO 3420
3440 IF KEZ = 11 THEN GOSUB 3310: GOTO 3420
3450 IF KEZ = 82 THEN RSZ(OCZ) = ABS (YH - YB(OCZ)): RJZ = RSZ(OCZ): XJ =
XB(OCZ): YJ = YB(OCZ): GOSUB 2760: GOTO 3040
3460 GOTO 3420
3500 ECZ = ECZ + 1: NSZ = 1: XE(ECZ) = XH: YE(ECZ) = YH
3510 GOSUB 3980: IF KEZ = 67 THEN NSZ = 0: ECZ = ECZ - 1: GOTO 3040
3520 IF ECZ > 1 THEN HCOLOR= 2: HPLLOT XE(ECZ - 1), YE(ECZ - 1) TO XE(ECZ
), YE(ECZ): HCOLOR= 3
3530 GOTO 3045
3550 NLZ = NLZ + 1: NSZ = 1
3560 GOSUB 3980: IF KEZ = 67 THEN NSZ = 0: NLZ = NLZ - 1: GOTO 3040
3570 IF NLZ > 0 THEN AA(NLZ) = SQR ((XM - XH) ^ 2 + (YM - YH) ^ 2): HPLLOT
XM, YM TO XH, YH: XM = XH: YM = YH
3580 GOTO 3045
3600 IF KIZ < = 0 THEN 3040
3610 KIZ = KIZ - 1: IF YIZ(KIZ + 1) < 1000 THEN HCOLOR= 0: HPLLOT XIZ(KIZ
+ 1), YIZ(KIZ + 1) TO XIZ(KIZ), INT ((YIZ(KIZ) / 1000 - INT (YIZ(KI
Z) / 1000)) * 1000 + .05): HCOLOR= 3
3620 GOTO 3040
3980 KEZ = PEEK ( - 16384): IF KEZ < 128 THEN 3980
3990 POKE - 16368, 0: KEZ = KEZ - 128: RETURN
8000 REM store input data
8010 TEXT : HOME : DO$ = CHR$ (4): VTAB 6: A$ = "Where do you want to sto
re the data ?": GOSUB 9000
8020 VTAB 9: HTAB 10: INPUT "FILE NAME ==> ": FI$: IF LEN (FI$) = 0 THEN
8020
8030 IF LEN (FI$) > 4 AND ( RIGHT$ (FI$, 4) = ".DAT" OR RIGHT$ (FI$, 4) =
".NET" OR RIGHT$ (FI$, 4) = ".PIC") THEN FI$ = LEFT$ (FI$, LEN (FI$
) - 4)
8040 FO$ = FI$ + ".DAT": PRINT DO$: PRINT DO$; "OPEN": FO$: PRINT DO$; "WRIT
E": FO$

```



```

8080 PRINT XO;",";YO;",";ECZ: FOR JJ = 1 TO ECZ: PRINT XE(JJ);",";YE(JJ)
: NEXT JJ: PRINT OCZ: FOR JJ = 1 TO OCZ: PRINT XB(JJ);",";YB(JJ);","
:RSZ(JJ): NEXT JJ
8090 PRINT NLZ: FOR JJ = 1 TO NLZ: PRINT AA(JJ): NEXT JJ: PRINT DO$;"CLO
SE";FO$: RETURN
8200 REM store the picture on screen
8210 IF KIZ < = 0 THEN RETURN
8220 DO$ = CHR$ (4):FO$ = FI$ + ".PIC"
8230 PRINT DO$: PRINT DO$"OPEN";FO$: PRINT DO$;"WRITE";FO$: PRINT KIZ: FOR
KK = 1 TO KIZ: PRINT XIZ(KK);",";YIZ(KK): NEXT KK: PRINT DO$;"CLOSE"
:FO$
8240 RETURN
8990 REM Centre string A$
9000 B = INT (20 - ( LEN (A$) / 2)): IF B = < 0 THEN B = 1
9010 HTAB B: PRINT A$: RETURN
9500 REM read stored data
9510 DO$ = CHR$ (4): TEXT : HOME : VTAB 5:A$ = "Where is the data stored
?": GOSUB 9000
9520 VTAB 8: HTAB 10: INPUT "FILE NAME ==> ";FI$: IF LEN (FI$) = 0 THEN
9520
9525 IF LEN (FI$) > 4 AND ( RIGHT$ (FI$,4) = ".DAT" OR RIGHT$ (FI$,4) =
".NET" OR RIGHT$ (FI$,4) = ".PIC") THEN FI$ = LEFT$ (FI$, LEN (FI$
) - 4)
9530 FO$ = FI$ + ".DAT"
9540 PRINT DO$: PRINT DO$;"READ";FO$: INPUT XO,YO,ECZ: FOR KK = 1 TO ECZ
: INPUT XE(KK),YE(KK): NEXT KK: INPUT OCZ: FOR KK = 1 TO OCZ: INPUT
XB(KK),YB(KK),RSZ(KK): NEXT KK
9550 INPUT NLZ: FOR KK = 1 TO NLZ: INPUT AA(KK): NEXT KK: PRINT DO$;"CLO
SE";FO$
9560 RETURN
10000 REM order the obstacles
10005 FOR JJ = 1 TO OCZ:DX = XB(JJ) - XO:DY = YB(JJ) - YO: GOSUB 10500:A
N(JJ) = SP: NEXT JJ
10010 FOR KK = (OCZ - 1) TO 1 STEP - 1: FOR JJ = 1 TO KK
10015 IF AN(JJ) < AN(JJ + 1) THEN TE = AN(JJ):AN(JJ) = AN(JJ + 1):AN(JJ +
1) = TE:TE = XE(JJ):XB(JJ) = XB(JJ + 1):XB(JJ + 1) = TE:TE = YB(JJ):
YB(JJ) = YB(JJ + 1):YB(JJ + 1) = TE:TE = RSZ(JJ):RSZ(JJ) = RSZ(JJ +
1):RSZ(JJ + 1) = TE
10020 NEXT JJ, KK: RETURN
10030 REM assign distances
10040 FOR KK = 1 TO OCZ:RD(KK) = SQR ((XB(KK) - XO) * (XB(KK) - XO) + (
YB(KK) - YO) * (YB(KK) - YO)): NEXT KK: RETURN
10500 REM gives theta from vector
10510 SP = ATN (DY / DX): IF DX < 0 THEN SP = PI + SP
10520 RETURN
10550 REM find convex polygon
10560 ZA = XB(1):ZB = XB(1): FOR KK = 2 TO OCZ: IF ZA < XB(KK) THEN ZA =
XB(KK)
10570 IF XB(KK) < ZB THEN ZB = XB(KK)
10580 NEXT KK:ZC = YB(1):ZD = YB(1): FOR KK = 2 TO OCZ: IF YB(KK) > ZC THEN
ZC = YB(KK)
10590 IF YB(KK) < ZD THEN ZD = YB(KK)
10600 NEXT KK:XC = (ZA + ZB) / 2:YC = (ZC + ZD) / 2:RR = ZA - XC: IF (ZC
- YC) > RR THEN RR = (ZC - YC)
10605 PRINT : PRINT : HTAB 6:A$ = "Type in the starting angle": GOSUB 90
00:A$ = "for circular scan.": GOSUB 9000: PRINT : HTAB 7: INPUT "Sta
rting angle in deg. = ";AG:AG = AG / BB
10610 RR = RR * 1.414:P3% = 0:S3%(P3%) = 0: FOR KK = 1 TO 2 * OCZ:ZT = AG
+ PI * (KK - 1) / OCZ:Z1 = XC + RR * COS (ZT):Z2 = YC + RR * SIN
(ZT)

```

```

10620 ZE = (XB(1) - Z1) * (XB(1) - Z1) + (YB(1) - Z2) * (YB(1) - Z2): ZG% =
1: FOR JJ = 1 TO OC%: ZF = (XB(JJ) - Z1) * (XB(JJ) - Z1) + (YB(JJ) -
Z2) * (YB(JJ) - Z2): IF ZF < ZE THEN ZE = ZF: ZG% = JJ
10630 NEXT JJ: IF S3%(P3%) = ZG% THEN 10650
10640 P3% = P3% + 1: S3%(P3%) = ZG%
10650 NEXT KK: IF S3%(P3%) = S3%(1) THEN P3% = P3% - 1
10660 RETURN
10700 REM form triangles
10710 IF OC% - P3% = 0 THEN 10750
10720 IF OC% - P3% = 1 THEN 10750
10750 TR% = OC% - 2: I% = 1: FOR KK = 1 TO P3%: S4%(KK) = S3%(KK): NEXT KK:
JJ = 0: P4% = P3%
10760 IF (I% + 2) < = P4% THEN JJ = JJ + 1: TQ$ = STR$ (S4%(I%)) + STR$
(S4%(I% + 1)) + STR$ (S4%(I% + 2)): GOSUB 10930: TR$(JJ) = TQ$: S4%(J
J) = S4%(I%): I% = I% + 2: GOTO 10760
10765 IF P4% = I% THEN 10780
10770 IF (P4% - I% = 1) AND (P4% > 2) THEN JJ = JJ + 1: TQ$ = STR$ (S4%(
I%)) + STR$ (S4%(I% + 1)) + STR$ (S4%(1)): GOSUB 10930: TR$(JJ) = T
Q$: S4%(JJ) = S4%(I%): GOTO 10780
10775 JJ = JJ + 1: TQ$ = STR$ (S4%(I%)) + STR$ (S4%(I% + 1)) + STR$ (S4
%(0)): GOSUB 10930: TR$(JJ) = TQ$: S4%(JJ) = S4%(I%)
10780 IF JJ < TR% THEN S4%(0) = S4%(P4%): P4% = JJ: I% = 1: GOTO 10760
10790 GOTO 10890
10800 KK = 0: SUZ = 1
10810 KK = KK + 1: IF (KK > OC%) OR (SUZ = 0) THEN 10840
10820 SUZ = 0: FOR JJ = 1 TO P3%: IF S3%(JJ) = KK THEN SUZ = 1
10830 NEXT JJ: GOTO 10810
10840 IF KK > OC% THEN 10750
10850 WH$ = STR$ (KK - 1): JJ = 0: FOR II = 1 TO (P3% - 1): JJ = JJ + 1: TQ
$ = STR$ (S3%(II)) + WH$ + STR$ (S3%(II + 1)): GOSUB 10930: TR$(JJ)
= TQ$: NEXT II
10860 JJ = JJ + 1: TQ$ = STR$ (S3%(P3%)) + WH$ + STR$ (S3%(1)): GOSUB 10
930: TR$(JJ) = TQ$: TR% = JJ: GOTO 10890
10890 RETURN
10900 REM verify edges
10910 IF VAL ( LEFT$ (EQ$, 1)) > VAL ( RIGHT$ (EQ$, 1)) THEN EQ$ = RIGHT$
(EQ$, 1) + LEFT$ (EQ$, 1)
10920 RETURN
10930 REM verify triangle
10940 FOR KK = 0 TO 2: TLX(KK) = VAL ( MID$ (TQ$, KK + 1, 1)): NEXT KK: IF
TLX(0) > TLX(1) THEN TE = TLX(0): TLX(0) = TLX(1): TLX(1) = TE
10950 IF TLX(1) > TLX(2) THEN TE = TLX(1): TLX(1) = TLX(2): TLX(2) = TE
10960 IF TLX(0) > TLX(1) THEN TE = TLX(0): TLX(0) = TLX(1): TLX(1) = TE
10970 TQ$ = STR$ (TLX(0)) + STR$ (TLX(1)) + STR$ (TLX(2)): RETURN
11000 REM identify edges and assign pointers
11010 WE$(0) = MID$ (WT$, 1, 2): WE$(1) = MID$ (WT$, 2, 2): WE$(2) = LEFT$ (
WT$, 1) + RIGHT$ (WT$, 1): FOR JJ = 0 TO 2: KK = 0
11020 KK = KK + 1: IF KK > ED% THEN 11050
11030 IF ED$(KK) = WE$(JJ) THEN 11060
11040 GOTO 11020
11050 ED% = ED% + 1: ED$(KK) = WE$(JJ): GOSUB 11600
11060 WJX(JJ) = KK: NEXT JJ
11070 TE = VAL ( LEFT$ (WT$, 1)): X1 = XB(TE): Y1 = YB(TE): TE = VAL ( MID$
(WT$, 2, 1)): X2 = XB(TE): Y2 = YB(TE): TE = VAL ( RIGHT$ (WT$, 1)): X3 =
XB(TE): Y3 = YB(TE)
11080 FOR JJ = 0 TO 2: X4 = HX(WJX(JJ)): Y4 = HY(WJX(JJ)): GOSUB 26150: WIX
(JJ) = SUZ: NEXT JJ
11090 IF WIX(0) > WIX(1) THEN TE = WIX(0): WIX(0) = WIX(1): WIX(1) = TE: TE
= WJX(0): WJX(0) = WJX(1): WJX(1) = TE
11100 IF WIX(1) > WIX(2) THEN TE = WIX(1): WIX(1) = WIX(2): WIX(2) = TE: TE
= WJX(1): WJX(1) = WJX(2): WJX(2) = TE

```

```

11110 IF WIX(0) > WIX(1) THEN TE = WIX(0):WIX(0) = WIX(1):WIX(1) = TE:TE
      = WJX(0):WJX(0) = WJX(1):WJX(1) = TE
11120 C1% = 0: FOR JJ = 0 TO 2: IF WIX(JJ) = 0 THEN C1% = C1% + 1
11130 NEXT JJ: IF C1% = 2 THEN 11150
11140 TMX(WJX(1)) = WJX(0):TNX(WJX(1)) = 0:TMX(WJX(2)) = WJX(0):TNX(WJX(2))
      = 0: GOTO 11150
11150 TMX(WJX(2)) = WJX(0):TNX(WJX(2)) = WJX(1)
11160 RETURN
11600 REM fill in head,tail and mid-points
11610 V1% = VAL ( LEFT$ (ED$(KK),1)):V2% = VAL ( RIGHT$ (ED$(KK),1)): IF
      XB(V2%) = XB(V1%) THEN M = 9.999E17: GOTO 11620
11615 M = (YB(V2%) - YB(V1%)) / (XB(V2%) - XB(V1%))
11620 M1 = RSX(V1%) / SQR (1 + M * M):X1 = XB(V1%) + SGN (XB(V2%) - XB(V1%))
      * M1:Y1 = M * (X1 - XB(V1%)) + YB(V1%):M2 = RSX(V2%) / SQR (1
      + M * M):X2 = XB(V2%) + SGN (XB(V1%) - XB(V2%)) * M2:Y2 = YB(V2%) +
      M * (X2 - XB(V2%)):X3 = (X1 + X2) / 2:Y3 = (Y1 + Y2) / 2
11625 EX(KK) = X3:EY(KK) = Y3
11630 X4 = XB(V1%) + M * M1:Y4 = YB(V1%) - (X4 - XB(V1%)) / M:X5 = XB(V2%)
      + M * M2:Y5 = YB(V2%) - (X5 - XB(V2%)) / M:M3 = (Y5 - Y4) / (X5 -
      X4):HX(KK) = - (Y4 - Y3 - X3 / M - M3 * X4) / (1 / M + M3):HY(KK) =
      Y3 - (HX(KK) - X3) / M
11640 X4 = XB(V1%) - M * M1:Y4 = YB(V1%) - (X4 - XB(V1%)) / M:X5 = XB(V2%)
      - M * M2:Y5 = YB(V2%) - (X5 - XB(V2%)) / M:M4 = (Y5 - Y4) / (X5 -
      X4):TX(KK) = - (Y4 - Y3 - X3 / M - M4 * X4) / (1 / M + M4):TY(KK) =
      Y3 - (TX(KK) - X3) / M
11650 IF (X0 - HX(KK)) ^ 2 + (Y0 - HY(KK)) ^ 2 > (X0 - TX(KK)) ^ 2 + (Y0
      - TY(KK)) ^ 2 THEN TE = HX(KK):HX(KK) = TX(KK):TX(KK) = TE:TE = HY(KK)
      :HY(KK) = TY(KK):TY(KK) = TE
11660 RETURN
26150 REM point inside triangle ?
26160 A = (Y4 - Y1):B = (X1 - X4):C = - B * Y1 - A * X1:L1 = A * X3 + B *
      Y3 + C:L2 = A * X2 + B * Y2 + C:C1% = 0: IF (L1 > 0 AND L2 > 0) OR (
      L1 < 0 AND L2 < 0) THEN C1% = 1
26170 A = (Y4 - Y2):B = (X2 - X4):C = - B * Y2 - A * X2:L1 = A * X3 + B *
      Y3 + C:L2 = A * X1 + B * Y1 + C:C2% = 0: IF (L1 > 0 AND L2 > 0) OR (
      L1 < 0 AND L2 < 0) THEN C2% = 1
26180 A = (Y4 - Y3):B = (X3 - X4):C = - B * Y3 - A * X3:L1 = A * X1 + B *
      Y1 + C:L2 = A * X2 + B * Y2 + C:C3% = 0: IF (L1 > 0 AND L2 > 0) OR (
      L1 < 0 AND L2 < 0) THEN C3% = 1
26190 IF (C1% AND C2%) OR (C2% AND C3%) OR (C1% AND C3%) THEN SUZ = 0: RETURN
26195 SUZ = 1: RETURN
41000 DIM EX(12),EY(12),ED$(12),HX(12),HY(12),TX(12),TY(12),TMX(12),TNX(
      12),RD(10),AN(10),TLX(2),WE$(2),WJX(2),WIX(2)
41010 RETURN
50000 REM print all information
50010 FOR II = 1 TO OC%: PRINT "XB(";II;") = ";XB(II);" YB(";II;") = ";Y
      B(II): NEXT II
50020 GET KE$: FOR II = 1 TO OC%: PRINT "RSX(";II;") = ";RSX(II): NEXT I
      I: GET KE$: FOR II = 1 TO ED%: PRINT "ED$(";II;") = ";ED$(II);" TMX(
      ";II;") = ";TMX(II);" TNX(";II;") = ";TNX(II): PRINT "HX(";II;") = "
      ;HX(II);" HY(";II;") = ";HY(II)
50030 PRINT "TX(";II;") = ";TX(II);" TY(";II;") = ";TY(II): PRINT "EX(";
      II;") = ";EX(II);" EY(";II;") = ";EY(II): NEXT II
50040 RETURN
50100 REM print input data
50110 TEXT : HOME : VTAB 5: HTAB 5: PRINT "Starting point : (";X0;",";Y0
      ;")": PRINT : HTAB 7: PRINT "Number of end points = ";EC%: FOR JJ =
      1 TO EC%: HTAB 7: PRINT "End point ";JJ;" at (";XE(JJ);",";YE(JJ);")
      ": NEXT JJ

```

```

50120 PRINT : HTAB 7: PRINT "Number of obstacles = ";OC%: FOR JJ = 1 TO
      OC%: HTAB 3: PRINT "Obstacle ";JJ;" at (";XB(JJ);",";YB(JJ);") with
      radius ";RSZ(JJ): NEXT JJ
50130 IF NL% > 0 THEN PRINT : HTAB 7: PRINT "Number of links = ";NL%: L =
      0: FOR JJ = 1 TO NL%: HTAB 5: PRINT "Length of link ";JJ;" = ";AA(JJ
      ): L = L + AA(JJ): NEXT JJ: HTAB 9: PRINT "Total length = ";L: RETURN
50200 REM plot the points
50210 HGR2 : HCOLOR= 3: GOSUB 50300
50220 FOR JJ = 1 TO OC%: GOSUB 50340: NEXT JJ
50290 RETURN
50300 HPLLOT 0,191 TO 0,0 TO 279,0
50310 HPLLOT 273,3 TO 279,10: HPLLOT 279,3 TO 273,10: HPLLOT 3,184 TO 7,188
      : HPLLOT 4,191 TO 9,184
50320 FOR JJ = 10 TO 270 STEP 10: HPLLOT JJ,0 TO JJ,3: NEXT : FOR JJ = 10
      TO 180 STEP 10: HPLLOT 0,JJ TO 3,JJ: NEXT
50330 RETURN
50340 FOR KK = 0 TO 360 STEP 10: X1 = XB(JJ) + RSZ(JJ) * COS (KK / BB): Y
      1 = YB(JJ) + RSZ(JJ) * SIN (KK / BB): HPLLOT XB(JJ),YB(JJ) TO X1,Y1:
      NEXT KK: RETURN

```

```

1  REM *****
2  REM *
3  REM * OBSTACLE AVOIDANCE *
4  REM * PROGRAM *
5  REM *
6  REM * Anupam Bagchi *
7  REM * Dec 1986 *
8  REM *
9  REM *****
20  DIM ED$(10),XH(30),YH(30),X(9,5),Y(9,5),XE(15),YE(15),XB(6),YB(6),S1%
    (6),S3%(5),S4%(5),LH(5),EX(10),EY(10),HX(10),HY(10),TX(10),TY(10),RS
    %(7),AN(7),RD(7),L2%(3),TMX(10),TNX(10),AA(6),XF(15),YF(15),MK%(6)
50  PI = 3.14159265:BB = 180 / PI:EPZ = 10
105 GOSUB 2100:RD(OC% + 1) = 1E20:RD(0) = 1E20
110 GOSUB 60000: POKE 216,0
150 HOME : VTAB 10: FOR KK = 1 TO NLZ: PRINT "Limit of Joint ";KK;" = ";
    : INPUT "~";JL(KK):JL(KK) = JL(KK) / BB: NEXT KK: GOTO 240
160 FOR KK = 1 TO P3% - 1: HPLLOT XB(S3%(KK)),YB(S3%(KK)) TO XB(S3%(KK +
    1)),YB(S3%(KK + 1)): NEXT KK: HPLLOT XB(S3%(P3%)),YB(S3%(P3%)) TO XB(
    S3%(1)),YB(S3%(1))
240 GG% = 1:PFZ = 0:EIZ = 0:EH% = 1:NP% = 5:NS% = 5:U2% = - 1:H% = 0
250 GOSUB 24000: GOSUB 28000
255 GOTO 250
260 GOSUB 50200: FOR HH = 1 TO ED%: HPLLOT TX(HH),TY(HH): GET KE$: HPLLOT
    TX(HH),TY(HH) TO HX(HH),HY(HH): NEXT HH
1000 END
2100 REM read network from disk
2110 TEXT : HOME : VTAB 6: PRINT "Which file do you want to operate upon
    ?";
2120 VTAB 9: HTAB 10: INPUT "FILE NAME ==> ";FI$: IF LEN (FI%) = 0 THEN
    2120
2130 IF LEN (FI%) > 4 AND ( RIGHT$ (FI$,4) = ".DAT" OR RIGHT$ (FI$,4) =
    ".NET" OR RIGHT$ (FI$,4) = ".PIC" OR RIGHT$ (FI$,4) = ".JNT") THEN
    FI$ = LEFT$ (FI$, LEN (FI%) - 4)
2140 F0$ = FI$ + ".NET"
2150 D0$ = CHR$ (4): PRINT D0$: PRINT D0$;"READ";F0$
2160 INPUT XO,YO,EC%: FOR JJ = 1 TO EC%: INPUT XE(JJ),YE(JJ): NEXT JJ
2170 INPUT OC%: FOR JJ = 1 TO OC%: INPUT XB(JJ),YB(JJ),RSX(JJ),RD(JJ),AN
    (JJ): NEXT JJ: INPUT NLZ:L = 0: FOR JJ = 1 TO NLZ: INPUT AA(JJ):L =
    L + AA(JJ): NEXT JJ
2180 INPUT ED%: FOR JJ = 1 TO ED%: INPUT ED$(JJ),TMX(JJ),TNX(JJ),HX(JJ),
    HY(JJ),TX(JJ),TY(JJ),EX(JJ),EY(JJ): NEXT JJ
2190 PRINT D0$;"CLOSE";F0$: RETURN
10500 REM gives slope in -90 to +270 range
10510 IF DX = 0 THEN SP = PI / 2: GOTO 10530
10520 SP = ATN (DY / DX)
10530 IF DX < 0 THEN SP = SP + PI
10540 RETURN
10600 REM gives slope in Quad 1 & 4
10610 IF DX = 0 THEN SP = PI / 2: GOTO 10630
10620 SP = ATN (DY / DX)
10630 RETURN
20000 REM find manipulator state
20010 DX = (XP - XO):DY = (YP - YO): GOSUB 10500
20015 IF SP > = AN(1) THEN NTX = 1: RETURN

```



```

20018 IF SP < AN(OCZ) THEN NTZ = OCZ + 1: RETURN
20020 NTZ = 2
20025 IF SP < AN(NTZ - 1) AND SP > AN(NTZ) THEN RETURN
20028 NTZ = NTZ + 1: GOTO 20025
20030 RETURN
20050 REM find index of reqd. edge
20060 JZ = 0
20070 JZ = JZ + 1: IF (ED$(JZ) = EQ$) OR (JZ > EDZ) THEN 20090
20080 GOTO 20070
20090 IF IZ > EDZ THEN EDZ = EDZ + 1: KK = EDZ: ED$(KK) = EQ$: GOSUB 11600

20095 RETURN
20100 REM assign markers to edges
20110 FOR JJ = 0 TO P1Z: MKZ(JJ) = 0: NEXT JJ: X4 = X0: Y4 = Y0: JJ = P1Z: P4
  Z = 0
20130 IF JJ = - 1 THEN 20190
20140 GOSUB 20300: SUZ = 0
20150 II = II + 1: IF (II > P4Z) OR (SUZ = 1) THEN 20170
20160 X3 = XB(S4Z(II)): Y3 = YB(S4Z(II)): R1 = RSZ(S4Z(II)): GOSUB 26000: GOTO
  20150
20170 IF SUZ = 1 THEN MKZ(JJ + 1) = 1: X4 = EX(JJ + 1): Y4 = EY(JJ + 1): P4
  Z = 0: JJ = JJ + 1
20180 JJ = JJ - 1: GOTO 20130
20190 RETURN
20300 REM update obstacle set
20310 V1Z = VAL ( LEFT$ (ED$(S1Z(JJ)), 1)): V2Z = VAL ( RIGHT$ (ED$(S1Z(J
  JJ)), 1)): II = 0
20320 II = II + 1: IF (II > P4Z) OR (S4Z(II) = V1Z) THEN 20350
20340 GOTO 20320
20350 IF II > P4Z THEN P4Z = P4Z + 1: S4Z(P4Z) = V1Z
20360 II = 0
20370 II = II + 1: IF (II > P4Z) OR (S4Z(II) = V2Z) THEN 20390
20380 GOTO 20370
20390 IF II > P4Z THEN P4Z = P4Z + 1: S4Z(P4Z) = V2Z
20395 RETURN
20500 REM find Bezier points within network
20510 KK = 0: D2Z = D2Z + 1: D1Z = 0: X(D1Z, D2Z) = X(L2Z(D2Z), D2Z - 1): Y(D1Z
  , D2Z) = Y(L2Z(D2Z), D2Z - 1): D1Z = D1Z + 1: X(D1Z, D2Z) = HX(S1Z(KK)): Y
  (D1Z, D2Z) = HY(S1Z(KK))
20520 KK = KK + 1: IF KK > P1Z THEN 20600
20530 IF 1 = MKZ(KK) THEN 20560
20535 IF P1Z < KK + 1 THEN 20560
20537 V1Z = VAL ( LEFT$ (ED$(S1Z(KK)), 1)): V2Z = VAL ( RIGHT$ (ED$(S1Z(K
  K)), 1)): DX = XB(V1Z) - XB(V2Z): DY = YB(V1Z) - YB(V2Z): GOSUB 10600: M
  1 = SP: DX = X(D1Z, D2Z) - EX(S1Z(KK)): DY = Y(D1Z, D2Z) - EY(S1Z(KK))
20540 GOSUB 10600: M1 = M1 - SP: LJ = (EX(S1Z(KK)) - EX(S1Z(KK + 1))) ^ 2 +
  (EY(S1Z(KK)) - EY(S1Z(KK + 1))) ^ 2: M = (EY(S1Z(KK)) - EY(S1Z(KK + 1
  ))) / (EX(S1Z(KK)) - EX(S1Z(KK + 1)))
20545 D1Z = D1Z + 1: X(D1Z, D2Z) = EX(S1Z(KK)) + SGN (EX(S1Z(KK + 1)) - EX
  (S1Z(KK))) * ABS ( COS (M1)) * SQR (LJ / (1 + M * M)): Y(D1Z, D2Z) =
  M * (X(D1Z, D2Z) - EX(S1Z(KK))) + EY(S1Z(KK))
20550 KK = KK + 1: IF KK > P1Z THEN 20600
20560 IF MKZ(KK) = 1 THEN 20580
20570 D1Z = D1Z + 1: X(D1Z, D2Z) = EX(S1Z(KK)): Y(D1Z, D2Z) = EY(S1Z(KK)): GOTO
  20550
20580 D1Z = D1Z + 1: X(D1Z, D2Z) = TX(S1Z(KK)): Y(D1Z, D2Z) = TY(S1Z(KK)): D1Z
  = D1Z + 1: X(D1Z, D2Z) = EX(S1Z(KK)): Y(D1Z, D2Z) = EY(S1Z(KK))
20590 L2Z(D2Z) = D1Z: D2Z = D2Z + 1: D1Z = 0: X(D1Z, D2Z) = EX(S1Z(KK)): Y(D1Z
  , D2Z) = EY(S1Z(KK)): D1Z = D1Z + 1: X(D1Z, D2Z) = HX(S1Z(KK)): Y(D1Z, D2Z
  ) = HY(S1Z(KK)): GOTO 20550

```



```

20600 L2%(D2%) = D1%: RETURN
20700 REM find nearest obstacle
20710 V1% = VAL ( LEFT$ (ED$(S1%(P1%)),1)):V2% = VAL ( RIGHT$ (ED$(S1%(P1%)),1)): IF YB(V1%) = Y3(V2%) THEN M = 9.999E17: GOTO 20720
20715 M = (XB(V1%) - XB(V2%)) / (Y3(V2%) - Y3(V1%))
20720 XW% = EX(S1%(P1%)) + SGN (X3 - EX(S1%(P1%))) * L / SQR (1 + M * M)
      :YW% = (M * (XW% - EX(S1%(P1%))) + EY(S1%(P1%)) + Y0) / 2
20725 XW% = X0 - L / 2:YW% = Y0 / 2
20730 RETURN
20800 REM solve dilemma with two pointers
20810 IF (X0 - EX(TM%(J%))) ^ 2 + (Y0 - EY(TM%(J%))) ^ 2 < (X0 - EX(TN%(J%))) ^ 2 + (Y0 - EY(TN%(J%))) ^ 2 THEN P1% = P1% + 1:S1%(P1%) = TM%(J%):J% = TM%(J%): RETURN
20820 P1% = P1% + 1:S1%(P1%) = TN%(J%):J% = TN%(J%): RETURN
21500 REM calculates maximum amplitude
21510 WA(0) = 0:WA(L2%(KK)) = 1:A = 0: FOR ZZ = 1 TO L2%(KK):A = A + SQR ((X(ZZ, KK) - X(ZZ - 1, KK)) ^ 2 + (Y(ZZ, KK) - Y(ZZ - 1, KK)) ^ 2): NEXT ZZ
21520 B = 0: FOR ZZ = 1 TO L2%(KK) - 1:B = B + SQR ((X(ZZ, KK) - X(ZZ - 1, KK)) ^ 2 + (Y(ZZ, KK) - Y(ZZ - 1, KK)) ^ 2):WA(ZZ) = B / A: NEXT ZZ
21530 C = 0: FOR ZZ = 1 TO L2%(KK): IF ABS (ZZ / L2%(KK) - WA(ZZ)) > ABS (C) THEN C = ZZ / L2%(KK) - WA(ZZ)
21540 NEXT ZZ: PRINT "Maximum amplitude = ";C: RETURN
23000 REM find correction factor
23010 PL = .5 * C * SIN (V * PI): PRINT "Correction factor = ";PL: RETURN

24000 REM find next end-effector position
24010 IF EIZ > 0 THEN 24100
24015 IF PEEK ( - 16384) > 127 THEN PRINT : PRINT CHR$ (7);"Jumping to next input point on request":GG% = PF%: POKE - 16384,0
24020 IF GG% < PF% THEN GG% = GG% + 1:XP = XF(GG%):YP = YF(GG%): RETURN

24030 IF EH% > EC% THEN PRINT DO$;"CLOSE":FO$: END
24040 PF% = SQR ((XE(EH%) - XE(EH% + 1)) ^ 2 + (YE(EH%) - YE(EH% + 1)) ^ 2) / 10: IF (XE(EH%) - XE(EH% + 1)) < 0 THEN 24055
24045 FOR KK = 1 TO PF%:XF(KK) = XE(EH%):YF(KK) = YE(EH%) + SGN (YE(EH% + 1) - YE(EH%)) * KK * 10: NEXT KK:XF(0) = XE(EH%):YF(0) = YE(EH%): GOTO 24080
24055 M4 = (YE(EH%) - YE(EH% + 1)) / (XE(EH%) - XE(EH% + 1))
24060 FOR KK = 1 TO PF%:XF(KK) = XE(EH%) + SGN (XE(EH% + 1) - XE(EH%)) * 10 * KK / SQR (1 + M4 * M4):YF(KK) = M4 * (XF(KK) - XE(EH%)) + YE(EH%): NEXT KK:XF(0) = XE(EH%):YF(0) = YE(EH%)
24080 EH% = EH% + 1:GG% = - 1: GOTO 24020
24100 IF EIZ = 2 THEN EG% = EG% + 1: IF EG% > PG% THEN EIZ = 0: PRINT CHR$ (7): PRINT "State change operation is over.": PRINT "Resuming operational path now.": PRINT :GG% = GG% - 1: GOTO 24010
24110 IF EIZ = 2 THEN XP = XH(EG%):YP = YH(EG%): RETURN
24120 Q1 = XP:Q2 = YP:Q3 = XP:Q4 = YP:S4%(1) = NP% - 1:P4% = 1:XC = (Q1 + XB(S4%(1))) / 2:YC = (Q2 + YB(S4%(1))) / 2: GOSUB 24500:EG% = 0:EIZ = 2: GOTO 24100
24460 REM tells theta from vector
24470 IF DX = 0 THEN SP = PI / 2: GOTO 24485
24475 SP = ATN (DY / DX): IF DX > 0 AND DY > 0 THEN RETURN
24478 IF DX < 0 THEN SP = SP + PI: RETURN
24480 IF DX > 0 AND DY < 0 THEN SP = 2 * PI + SP
24490 RETURN
24500 REM how to change state ?
24505 PRINT CHR$ (7): PRINT "Going for a STATE CHANGE now.": PRINT "Approaching obstacle ";S4%(P4%): PRINT
24510 PG% = 0:DX = (Q1 - XB(S4%(1))):DY = (Q2 - YB(S4%(1))): IF DY = 0 THEN M1 = 9.999E17: GOTO 24530

```

```

24520 M1 = - (DX / DY)
24530 X1 = XB(S4%(1)) + (RSZ(S4%(1)) + EPX) / SQR (1 + M1 * M1):Y1 = YB(
    S4%(1)) + M1 * (X1 - XB(S4%(1)))
24535 REM X2=XB(S4%(1))-(RSZ(S4%(1))+EPX)/SQR(1+M1*M1):Y2=YB(S4%(1))+M1*
    (X2-XB(S4%(1))):IF(X2-XC)^2+(Y2-YC)^2<(X1-XC)^2+(Y1-YC)^2THENX1=X2:Y
    1=Y2
24537 X2 = XB(S4%(1)) - (RSZ(S4%(1)) + EPX) / SQR (1 + M1 * M1):Y2 = YB(
    S4%(1)) + M1 * (X2 - XB(S4%(1))): IF Y2 < Y1 THEN X1 = X2:Y1 = Y2
24540 W1 = Q1:W2 = Q2:W3 = X1:W4 = Y1: GOSUB 24800:II = 1
24550 II = II + 1: IF II > P4% THEN 24650
24560 DY = YB(S4%(II)) - YB(S4%(II - 1)):DX = XB(S4%(II)) - XB(S4%(II - 1
    )): IF DY = 0 THEN M1 = 9.999E17: GOTO 24580
24570 M1 = - (DX / DY)
24580 X2 = XB(S4%(II - 1)) + (RSZ(S4%(II - 1)) + EPX) / SQR (1 + M1 * M1
    ):Y2 = YB(S4%(II - 1)) + M1 * (X2 - XB(S4%(II - 1)))
24585 X3 = XB(S4%(II - 1)) - (RSZ(S4%(II - 1)) + EPX) / SQR (1 + M1 * M1
    ):Y3 = YB(S4%(II - 1)) + M1 * (X3 - XB(S4%(II - 1)))
24590 IF (X3 - XC) ^ 2 + (Y3 - YC) ^ 2 > (X2 - XC) ^ 2 + (Y2 - YC) ^ 2 THEN
    X2 = X3:Y2 = Y3
24600 W1 = X1:W2 = Y1:W3 = X2:W4 = Y2:W5 = XB(S4%(II - 1)):W6 = YB(S4%(II
    - 1)):R1 = RSZ(S4%(II - 1)): GOSUB 24860
24610 X1 = XB(S4%(II)) + (RSZ(S4%(II)) + EPX) / SQR (1 + M1 * M1):Y1 = Y
    B(S4%(II)) + M1 * (X1 - XB(S4%(II)))
24615 X3 = XB(S4%(II)) - (RSZ(S4%(II)) + EPX) / SQR (1 + M1 * M1):Y3 = Y
    B(S4%(II)) + M1 * (X3 - XB(S4%(II)))
24620 IF (X3 - XC) ^ 2 + (Y3 - YC) ^ 2 > (X1 - XC) ^ 2 + (Y1 - YC) ^ 2 THEN
    X1 = X3:Y1 = Y3
24630 W1 = X2:W2 = Y2:W3 = X1:W4 = Y1: GOSUB 24800
24640 GOTO 24550
24650 DY = Q4 - YB(S4%(II - 1)):DX = Q3 - XB(S4%(II - 1)): IF DY = 0 THEN
    M1 = 9.999E17: GOTO 24670
24660 M1 = - (DX / DY)
24670 X2 = XB(S4%(II - 1)) - (RSZ(S4%(II - 1)) + EPX) / SQR (1 + M1 * M1
    ):Y2 = YB(S4%(II - 1)) + M1 * (Y2 - XB(S4%(II - 1)))
24675 REM X3=XB(S4%(II-1))+(RSZ(S4%(II-1))+EPX)/SQR(1+M1*M1):Y3=YB(S4%(I
    I-1))+M1*(X3-XB(S4%(II-1))):IF(X3-XC)^2+(Y3-YC)^2<(X2-XC)^2+(Y2-YC)^
    2THENX2=X3:Y2=Y3
24680 X3 = XB(S4%(II - 1)) + (RSZ(S4%(II - 1)) + EPX) / SQR (1 + M1 * M1
    ):Y3 = YB(S4%(II - 1)) + M1 * (X3 - XB(S4%(II - 1))): IF Y3 > Y2 THEN
    X2 = X3:Y2 = Y3
24690 W1 = X1:W2 = Y1:W3 = X2:W4 = Y2:W5 = XB(S4%(II - 1)):W6 = YB(S4%(II
    - 1)):R1 = RSZ(S4%(II - 1)): GOSUB 24860
24700 W3 = Q3:W4 = Q4:W1 = X2:W2 = Y2: GOSUB 24800: RETURN
24800 REM straight line segmentation
24810 IF (W1 - W3) = 0 THEN M3 = 9.999E17: GOTO 24830
24820 M3 = (W2 - W4) / (W1 - W3)
24830 NZ = SQR ((W1 - W3) ^ 2 + (W2 - W4) ^ 2) / 10: FOR KK = 1 TO NZ:PG
    % = PG% + 1:XH(PG%) = W1 + SGN (W3 - W1) * 10 * KK / SQR (1 + M3 *
    M3):YH(PG%) = M3 * (XH(PG%) - W1) + W2: NEXT KK
24840 RETURN
24860 REM circular segmentation
24870 DX = (W1 - W5):DY = (W2 - W6): GOSUB 10500:A = SP + 4 * PI:DX = (W3
    - W5):DY = (W4 - W6): GOSUB 10500:B = SP + 4 * PI
24875 PRINT "A = "; INT (A * BB); " deg": PRINT "B = "; INT (B * BB); " de
    g"
24880 NN = (ABS (B - A)) * (R1 + EPX) / 10:PD = 10 / (R1 + EPX): FOR KK =
    1 TO NN:PG% = PG% + 1:XH(PG%) = (R1 + EPX) * COS (A - KK * PD) + W5
    :YH(PG%) = (R1 + EPX) * SIN (A - KK * PD) + W6: NEXT KK
24890 RETURN

```

```

25000 REM validate set
25010 OF = SQR ((XD - XP) ^ 2 + (YD - YP) ^ 2): FOR KK = 1 TO OC%:OD = SQR
  ((XD - XB(KK)) ^ 2 + (YD - YB(KK)) ^ 2): IF OD - RS%(KK) > = L THEN
  TAZ%(KK) = 0: GOTO 25030
25020 OE = SQR ((XP - XB(KK)) ^ 2 + (YD - YB(KK)) ^ 2): IF OD + OE - 2 *
  RS%(KK) > = L THEN TAZ%(KK) = 0: GOTO 25030
25025 TAZ%(KK) = 1
25030 GOSUB 25050: NEXT KK: RETURN
25050 REM eliminate all beyond E
25060 IF DT(KK) > OF + RS%(KK) THEN TAZ%(KK) = 0
25070 RETURN
25490 IF (X3 - XC) ^ 2 + (Y3 - YC) ^ 2 > (X2 - XC) ^ 2 + (Y2 - YC) ^ 2 THEN
  X2 = X3:Y2 = Y3
26000 REM line cuts circle ?
26010 IF ((X1 - X3) ^ 2 + (Y1 - Y3) ^ 2) < R1 * R1 OR ((X2 - X3) ^ 2 + (
  Y2 - Y3) ^ 2) < R1 * R1 THEN SUZ% = 1: RETURN
26020 M = (Y2 - Y1) / (X2 - X1):X4 = (Y3 - Y1 + M * X1 + X3 / M) / (M + 1
  / M):Y4 = Y1 + M * (X4 - X1):PD = (X3 - X4) ^ 2 + (Y3 - Y4) ^ 2
26030 IF PD > R1 * R1 THEN SUZ% = 0: RETURN
26040 DX = X1 - X3:DY = Y1 - Y3: GOSUB 24470:M1 = SP:DX = X2 - X3:DY = Y2
  - Y3: GOSUB 24470:M2 = ABS (M1 - SP): IF M2 > PI THEN M2 = 2 * PI -
  M2
26050 IF M2 < PI / 2 THEN SUZ% = 0: RETURN
26060 SUZ% = 1: RETURN
26150 REM point inside triangle ?
26160 A = (Y4 - Y1):B = (X1 - X4):C = - B * Y1 - A * X1:L1 = A * X3 + B *
  Y3 + C:L2 = A * X2 + B * Y2 + C:C1% = 0: IF (L1 > 0 AND L2 > 0) OR (
  L1 < 0 AND L2 < 0) THEN C1% = 1
26170 A = (Y4 - Y2):B = (X2 - X4):C = - B * Y2 - A * X2:L1 = A * X3 + B *
  Y3 + C:L2 = A * X1 + B * Y1 + C:C2% = 0: IF (L1 > 0 AND L2 > 0) OR (
  L1 < 0 AND L2 < 0) THEN C2% = 1
26180 A = (Y4 - Y3):B = (X3 - X4):C = - B * Y3 - A * X3:L1 = A * X1 + B *
  Y1 + C:L2 = A * X2 + B * Y2 + C:C3% = 0: IF (L1 > 0 AND L2 > 0) OR (
  L1 < 0 AND L2 < 0) THEN C3% = 1
26190 IF (C1% AND C2%) OR (C2% AND C3%) OR (C1% AND C3%) THEN SUZ% = 0: RETUR
26195 SUZ% = 1: RETURN
28000 REM solves manipulator configuration
28005 HZ = HZ + 1:CZ = 1:FZ = 0: PRINT : HTAB 10: PRINT "Iteration number
  ";HZ: PRINT : IF SQR ((XP - XO) ^ 2 + (YP - YO) ^ 2) > L THEN PRINT
  "Point (" INT (XP)," INT (YP))" is inaccessible.": RETURN
28010 X1 = XP:Y1 = YP:X(0,0) = XP:Y(0,0) = YP: GOSUB 20000: PRINT "Nat.St
  ate = ";NTX: GOSUB 28500:NSX = NTX: IF NPX = 1 OR NPX > OC% THEN GOSUB
  35000: RETURN
28020 NR$ = STR$ (NPX - 1) + STR$ (NPX): PRINT "Present first edge = ";
  NR$: IF NPX < > NSX THEN GOSUB 30500: GOTO 28050
28030 X1 = XO:Y1 = YO:X2 = XB(NSX):Y2 = YB(NSX):X3 = XB(NSX - 1):Y3 = YB(
  NSX - 1):X4 = XP:Y4 = YP: GOSUB 26150: IF SUZ% = 1 THEN GOSUB 30200:
  GOTO 28050
28040 GOSUB 30000
28050 IF (II = 0 AND FZ = 0) THEN 28200
28060 FZ = 0:CZ = CZ + 1: GOSUB 31300:D1% = 0:D2% = 0:KK = 0: GOSUB 20530
  : GOSUB 31000: GOSUB 32000: GOSUB 33000
28070 IF (II = 0 AND FZ = 0) THEN 28200
28080 FZ = 0:CZ = CZ + 1: GOSUB 31500:D1% = 0:D2% = 0:KK = 0: GOSUB 20530
  : GOSUB 31000: GOSUB 32000: GOSUB 33000
28100 IF (II < > 0 OR FZ = 1) AND EIX = 0 THEN EIX = 1
28200 RETURN
28500 REM is calculation necessary ?
28510 IF NTX = NSX THEN 28550

```



```

28520 A = NTX: IF RD(NSX) < RD(A) THEN A = NSX
28525 PRINT "Crossing a ray at this stage.": PRINT "Previous natural sta
    te = ";NSX: PRINT "Present state = ";NPX: PRINT "Nearest obstacle =
    ";A

28530 IF (X0 - XP) ^ 2 + (Y0 - YP) ^ 2 < RD(A) * RD(A) THEN NPX = NTX:U1
    % = 1:U2% = - 1: RETURN
28540 U1% = 0: RETURN
28550 IF NSX < > NPX THEN U1% = 0: RETURN
28560 X1 = X0:Y1 = Y0:X2 = XB(NPX - 1):Y2 = YB(NPX - 1):X3 = XB(NPX):Y3 =
    YB(NPX): GOSUB 26150: IF SU% = U2% THEN U1% = 0: RETURN
28570 U2% = SU%:U1% = 1: RETURN
29000 IF X = 0 OR X = 1 THEN FC = 1: RETURN
29010 FC = X: FOR I = X - 1 TO 2 STEP - 1:FC = FC * I: NEXT I: RETURN
29100 REM n C r
29110 X = NN: GOSUB 29000:NC = FC:X = RR: GOSUB 29000:NC = NC / FC:X = NN
    - RR: GOSUB 29000:NC = NC / FC: RETURN
29120 REM Modified Composite Simpson's Rule
29130 FOR UU = 1 TO N:W(UU) = (UU - 1) / (N - 1):F(UU) = SQR (DX(UU) *
    DX(UU) + DY(UU) * DY(UU)): NEXT UU:B = 1:A = 0
29140 H = (B - A) / (N - 1)
29150 N2 = N - 2:Q = 0
29160 FOR I = 2 TO N2
29170 I0 = I - 1:I1 = I + 1:I2 = I + 2
29180 B1 = (F(I) - F(I0)) / (W(I) - W(I0))
29190 B2 = (F(I1) - F(I)) / (W(I1) - W(I))
29200 B3 = (F(I2) - F(I1)) / (W(I2) - W(I1))
29210 C1 = (B2 - B1) / (W(I1) - W(I0))
29220 C2 = (B3 - B2) / (W(I2) - W(I1))
29230 D = (C2 - C1) / (W(I2) - W(I0))
29240 IF I = 2 THEN 29270
29250 IF I = N2 THEN 29280
29260 J = I:J1 = I1: GOTO 29290
29270 J = 1:J1 = 3: GOTO 29290
29280 J = N2:J1 = N
29290 ZO = (W(J1) + W(J)) / 2
29300 FO = F(I0) + (ZO - W(I0)) * (B1 + (ZO - W(I)) * (C1 + (ZO - W(I1)) *
    D))
29310 Q = Q + (W(J1) - W(J)) * (F(J1) + F(J) + 4 * FO) / 6
29320 NEXT I
29330 PRINT : HTAB 5: PRINT "VALUE OF THE INTEGRAL = ";Q
29340 RETURN
29390 REM find Bezier Point
29420 XG = 0:YG = 0:NN = L2X(KK): FOR RR = 0 TO L2X(KK): GOSUB 29110:PQ =
    INT (NC) * ((1 - U) ^ (NN - RR)) * (U ^ RR):XG = XG + X(RR,KK) * PQ
    :YG = YG + Y(RR,KK) * PQ: NEXT RR: RETURN
29490 REM length by integration
29500 N = 5
29510 FOR UU = 2 TO (N - 1):U = (UU - 1) / (N - 1):DX(UU) = 0:DY(UU) = 0
    :NN = L2X(KK): FOR RR = 0 TO L2X(KK): GOSUB 29110
29520 PQ = INT (NC) * ((1 - U) ^ (NN - RR - 1)) * (U ^ (RR - 1)) * (RR -
    NN * U):DX(UU) = DX(UU) + X(RR,KK) * PQ:DY(UU) = DY(UU) + Y(RR,KK) *
    PQ: NEXT RR: NEXT UU
29530 DX(1) = NN * (X(1,KK) - X(0,KK)):DY(1) = NN * (Y(1,KK) - Y(0,KK)):D
    X(N) = NN * (X(NN,KK) - X(NN - 1,KK)):DY(N) = NN * (Y(NN,KK) - Y(NN -
    1,KK))
29540 GOSUB 29120: RETURN
30000 REM point within range and outside edge
30005 PRINT "Point WITHIN range and OUTSIDE"
30010 EQ$ = NR$: GOSUB 20050:P1% = - 1:D1% = 0:D2% = 0: GOTO 30080
30015 ND = (TX(JX) - X0) ^ 2 + (TY(JX) - Y0) ^ 2:P1% = - 1:D1% = 0:D2% =

```

```

30020 NC = (X0 - XP) * (X0 - XP) + (Y0 - YP) * (Y0 - YP): IF NC > ND THEN
    D1% = D1% + 1: X(D1%, D2%) = TX(J%): Y(D1%, D2%) = TY(J%): GOTO 30040
30030 D1% = D1% + 1: X(D1%, D2%) = EX(J%): Y(D1%, D2%) = EY(J%): GOTO 30080
30040 X1 = XP: Y1 = YP: X2 = X(D1%, D2%): Y2 = Y(D1%, D2%): V1% = VAL ( LEFT$
    (NR$, 1)): X3 = XB(V1%): Y3 = YB(V1%): R1 = RSZ(V1%): GOSUB 20000: IF SU
    % = 1 THEN 30060
30050 V1% = VAL ( RIGHT$ (NR$, 1)): X3 = XB(V1%): Y3 = YB(V1%): R1 = RSZ(V1%
    ): GOSUB 20000: IF SU% = 0 THEN 30080
30060 NG = RD(NSZ) + RSZ(NSZ): IF RD(NSZ - 1) + RSZ(NSZ - 1) > NG THEN NG
    = RD(NSZ - 1) + RSZ(NSZ - 1)
30070 NH = (AN(NSZ) + AN(NSZ - 1)) / 2: D1% = D1% + 1: X(D1%, D2%) = X(D1% -
    1, D2%): Y(D1%, D2%) = Y(D1% - 1, D2%): X(D1% - 1, D2%) = X0 + NG * COS (
    NH): Y(D1% - 1, D2%) = Y0 + NG * SIN (NH)
30080 P1% = P1% + 1: S1%(P1%) = J%: L2%(D2%) = D1%
30090 IF TM%(J%) < > 0 AND TN%(J%) = 0 THEN P1% = P1% + 1: S1%(P1%) = TM
    %(J%): J% = TM%(J%): GOTO 30090
30093 IF TM%(J%) = 0 THEN 30100
30095 GOSUB 20800: GOTO 30090
30100 IF P1% = 0 THEN 30310
30103 GOSUB 20100: KK = 0: GOSUB 20530
30105 GOSUB 31000
30110 GOSUB 32000
30180 GOSUB 33000
30190 RETURN
30200 REM point within range and inside edge
30205 PRINT "Point WITHIN range and INSIDE edge"
30210 EQ$ = NR$: GOSUB 20050
30220 IF TM%(J%) = 0 THEN P1% = - 1: GOTO 30330
30230 J% = TM%(J%): X1 = X0: Y1 = Y0: V1% = VAL ( LEFT$ (ED$(J%), 1)): X2 = X
    B(V1%): Y2 = YB(V1%): V2% = VAL ( RIGHT$ (ED$(J%), 1)): X3 = XB(V2%): Y3
    = YB(V2%): X4 = XP: Y4 = YP: GOSUB 20150: IF SU% = 0 THEN 30240
30235 GOTO 30220
30240 P1% = 0: S1%(P1%) = J%
30280 IF TM%(J%) < > 0 AND TN%(J%) = 0 THEN P1% = P1% + 1: S1%(P1%) = TM
    %(J%): J% = TM%(J%): GOTO 30280
30290 IF TM%(J%) = 0 THEN 30310
30300 GOSUB 20800: GOTO 30280
30310 IF P1% = 0 THEN V1% = VAL ( LEFT$ (ED$(S1%(P1%)), 1)): V2% = VAL (
    RIGHT$ (ED$(S1%(P1%)), 1)): A = YB(V1%) - YB(V2%): B = XB(V2%) - XB(V1
    %): C = - YB(V2%) * B - XB(V2%) * A: PD = ABS (A * XP + B * YP + C) /
    SQR (A * A + B * B): IF PD < 2 * AA(NLZ) THEN 30330
30320 GOSUB 20100: D1% = 0: D2% = 0: KK = 0: GOSUB 20530: GOSUB 31000: GOTO
    30110
30330 PRINT "Extending arrow at edge "; ED$(J%); " for a point."
30340 D2% = 0: L2%(D2%) = 1: LH(0) = 0: M = (TY(J%) - HY(J%)) / (TX(J%) - HX
    (J%)): PQ = (HX(J%) - TX(J%)) ^ 2 + (HY(J%) - TY(J%)) ^ 2: IF P1% = 0
    THEN X(L2%(D2%), D2%) = EX(S1%(P1%)) + SGN (X0 - X(0, 0)) * SQR (6 *
    PQ / (1 + M * M)): GOTO 30350
30345 X(L2%(D2%), D2%) = X(0, 0) + SGN (X0 - X(0, 0)) * SQR (6 * PQ / (1 +
    M * M))
30350 Y(L2%(D2%), D2%) = M * (X(L2%(D2%), D2%) - X(0, 0)) + Y(0, 0): P1% = 0: S
    1%(P1%) = J%: GOTO 30110
30500 REM point not in natural state
30505 PRINT "Point is BEYOND range"
30510 EQ$ = NR$: GOSUB 20050: P1% = - 1: D1% = 0: D2% = 0: IF NP% < NS% THEN
    V1% = NP%: GOTO 30530
30520 V1% = NP% - 1
30530 NH = AN(V1%): NG = RD(V1%) + 3 * RSZ(V1%): X1 = X0 + NG * COS (NH): Y
    1 = Y0 + NG * SIN (NH): NC = (X0 - XP) ^ 2 + (Y0 - YP) ^ 2: ND = RD(V
    1%) + RSZ(V1%)

```

```

30540 IF NC > ND THEN 30590
30550 X2 = XP:Y2 = YP:X3 = XB(V1%):Y3 = YB(V1%):R1 = RSZ(V1%): GOSUB 2600
0: IF SUZ = 1 THEN 30580
30560 V2% = NSZ - 1: IF V2% < > V1% THEN X3 = XB(V2%):Y3 = YB(V2%):R1 =
RSZ(V2%): GOSUB 26000: IF SUZ = 1 THEN 30580
30570 V2% = NSZ: IF V2% < > V1% THEN X3 = XB(V2%):Y3 = YB(V2%):R1 = RSZ(
V2%): GOSUB 26000: IF SUZ = 0 THEN 30590
30580 NH = (AN(NSZ) + AN(NSZ - 1)) / 2:D1% = D1% + 1:X(D1%,0) = X0 + NG *
COS (NH):Y(D1%,0) = Y0 + NG * SIN (NH)
30590 D1% = D1% + 1:X(D1%,0) = X1:Y(D1%,0) = Y1:X2 = EX(J%):Y2 = EY(J%):X
3 = XB(V1%):Y3 = YB(V1%):R1 = RSZ(V1%): GOSUB 26000: IF SUZ = 1 THEN
NH = (AN(NP%) + AN(NP% - 1)) / 2:D1% = D1% + 1:X(D1%,0) = X0 + NG *
COS (NH):Y(D1%,0) = Y0 + NG * SIN (NH)
30600 D1% = D1% + 1:X(D1%,0) = TX(J%):Y(D1%,0) = TY(J%):D1% = D1% + 1:X(D
1%,0) = EX(J%):Y(D1%,0) = EY(J%):P1% = P1% + 1:S1%(P1%) = J%:L2%(D2%
) = D1%: IF U1% = 1 THEN 30620
30610 GOTO 30090
30620 KK = 0: GOSUB 29500:LH(KK) = Q: GOTO 30110
31000 REM integrate the Bezier Curves
31010 IF D2% = 0 THEN LH(D2%) = 0: RETURN
31020 FOR KK = 0 TO D2% - 1: GOSUB 29500:LH(KK) = Q: NEXT KK: RETURN
31100 REM find collision cases if any
31110 II = NL%
31120 IF II = 0 THEN 31190
31130 JJ = 1
31140 IF JJ > OC% THEN 31180
31160 X1 = XK(II):Y1 = YK(II):X2 = XK(II - 1):Y2 = YK(II - 1):X3 = XB(JJ)
:Y3 = YB(JJ):R1 = RSZ(JJ): GOSUB 26000: IF SUZ = 1 THEN PRINT "Link
";II;" collides with obstacle ";JJ: GOTO 31190
31170 JJ = JJ + 1: GOTO 31140
31180 II = II - 1: GOTO 31120
31190 RETURN
31300 REM find edge of collision
31310 KK = P1%:NC = 1.0E20:II = - 1
31320 IF KK < 0 THEN 31360
31330 V1% = VAL ( LEFT$ (ED$(S1%(KK)),1)):V2% = VAL ( RIGHT$ (ED$(S1%(K
K)),1)): IF V1% = JJ OR V2% = JJ THEN 31350
31340 KK = KK - 1: GOTO 31320
31350 ND = (XB(JJ) - EX(S1%(KK))) ^ 2 + (YB(JJ) - EY(S1%(KK))) ^ 2: IF ND
< = NC THEN II = KK:NC = ND
31355 GOTO 31340
31360 IF II < 0 THEN GOSUB 31500: RETURN
31370 MK%(II) = 1: RETURN
31500 REM mark all edges this time
31520 FOR II = 0 TO P1%:MK%(II) = 1: NEXT II
31570 RETURN
32000 REM find last compensatory length
32010 LE = 0: FOR KK = 0 TO D2% - 1:LE = LE + LH(KK): NEXT KK:LR = L - LE
32020 L2%(D2%) = L2%(D2%) + 1:X(L2%(D2%),D2%) = X0:Y(L2%(D2%),D2%) = Y0
32030 KK = D2%: GOSUB 29500: IF ABS (LR - Q) < 20 THEN LH(KK) = Q: RETURN
32040 NC = Q:V1% = VAL ( LEFT$ (ED$(S1%(P1%)),1)):V2% = VAL ( RIGHT$ (E
D$(S1%(P1%)),1)):DY = YB(V1%) - YB(V2%):DX = XB(V1%) - XB(V2%): GOSUB
24470
32043 GZ = 2:RX = 0
32045 M1 = SP:DX = X0 - EX(S1%(P1%)):DY = Y0 - EY(S1%(P1%)): GOSUB 24470:
M = (M1 + 3 * SP) / 4:M1 = TAN (M + PI / 2):M2 = (YB(V2%) - Y0) / (
XB(V2%) - X0)
32050 L2%(D2%) = L2%(D2%) + 1:X(L2%(D2%),D2%) = X0:Y(L2%(D2%),D2%) = Y0

```



```

32055 RZ = RZ + 1: X(L2%(D2%) - 1, D2%) = EX(S1%(P1%)) - GZ * ABS (NC - LR
) / SQR (1 + M1 * M1): Y(L2%(D2%) - 1, D2%) = M1 * (X(L2%(D2%) - 1, D2
%) - EX(S1%(P1%))) + EY(S1%(P1%))
32060 A = EX(S1%(P1%)) + GZ * ABS (NC - LR) / SQR (1 + M1 * M1): B = M1 *
(A - EX(S1%(P1%))) + EY(S1%(P1%))
32065 IF (A - XK(1)) ^ 2 + (B - YK(1)) ^ 2 < (X(L2%(D2%) - 1, D2%) - XK(1
)) ^ 2 + (Y(L2%(D2%) - 1, D2%) - YK(1)) ^ 2 THEN :X(L2%(D2%) - 1, D2%)
= A: Y(L2%(D2%) - 1, D2%) = B
32070 GOSUB 29500: IF ABS (Q - LR) > 10 AND RZ < 10 THEN GZ = GZ * LR /
Q: GOTO 32055
32200 LH(KK) = Q: RETURN
33000 REM find Joint Coordinates
33005 PRINT "Number of curves : "; D2% + 1: PRINT "Stack : "; FOR ZZ = 0 TO
P1%: HTAB 9: PRINT ED$(S1%(ZZ)); " "; MK%(ZZ): NEXT ZZ
33007 IF HZ = 1 THEN GOSUB 20700
33010 F0$ = FI$ + ".JNT": LE = 0: KK = 0: GOSUB 21500: LJ = LH(KK): II = NL%:
XK(II) = XP: YK(II) = YP: XL(II) = XP: YL(II) = YP: XK(0) = XO: YK(0) = Y
O: XL(0) = XO: YL(0) = YO
33020 IF II = 1 THEN 33080
33030 LE = LE + AA(II)
33040 IF LE > LJ THEN 33070
33050 V = (LE - LJ + LH(KK)) / LH(KK): GOSUB 23000: U = V + PL: GOSUB 2942
O: XL(II - 1) = XG: YL(II - 1) = YG
33060 II = II - 1: GOTO 33020
33070 KK = KK + 1: GOSUB 21500: LJ = LJ + LH(KK): GOTO 33040
33080 II = NL%
33090 XM = XP: YM = YP: XG = XL(II - 1): YG = YL(II - 1)
33100 GOSUB 33400
33120 XK(II - 1) = XI: YK(II - 1) = YI
33130 II = NL% - 1
33140 IF II = 2 THEN 33200
33150 XM = XK(II): YM = YK(II): XG = XL(II - 1): YG = YL(II - 1): GOSUB 3340
O: DX = XK(II) - XI: DY = YK(II) - YI: GOSUB 10500: C = SP - M1: IF ABS
(C) > PI THEN C = C - SGN (C) * 2 * PI
33155 IF ABS (C) < . = JL(II + 1) THEN M1 = SP: GOTO 33170
33160 M1 = M1 - SGN (C) * JL(II + 1): PRINT "Joint "; II + 1: " is being p
ut to the limit": M = TAN (M1): GOSUB 33430
33170 XK(II - 1) = XI: YK(II - 1) = YI: II = II - 1: GOTO 33140
33200 XM = XK(2): YM = YK(2): D = (XM - XO) ^ 2 + (YM - YO) ^ 2: IF SQR (D
) > AA(1) + AA(2) THEN GOSUB 33500: GOTO 33260
33210 K1 = (D + AA(1) * AA(1) - AA(2) * AA(2)) / (2 * AA(1)): K2 = K1 / SQR
(D): IF (XM - XO) = 0 THEN AL = PI / 2: GOTO 33230
33220 AL = ATN ((YM - YO) / (XM - XO)): IF XM < XO THEN AL = AL + PI
33230 TH = AL + ATN (SQR (1 - K2 * K2) / K2): XI = XO + AA(1) * COS (TH
): YI = YO + AA(1) * SIN (TH)
33240 TH = AL - ATN (SQR (1 - K2 * K2) / K2): XK = XO + AA(1) * COS (TH
): YK = YO + AA(1) * SIN (TH): IF (XK - XWZ) ^ 2 + (YK - YWZ) ^ 2 <
(XI - XWZ) ^ 2 + (YI - YWZ) ^ 2 THEN XI = XK: YI = YK
33250 XK(1) = XI: YK(1) = YI
33260 IF FZ = 1 THEN 33290
33265 GOSUB 31100: IF II < > 0 THEN 33290
33270 PRINT D0$: PRINT D0$: "WRITE": F0$: FOR ZZ = NL% TO 1 STEP - 1: PRINT
XK(ZZ); ", "; YK(ZZ): NEXT ZZ: PRINT XO; ", "; YO: PRINT D0$
33280 FOR ZZ = NL% TO 2 STEP - 1: PRINT "HPOINT "; INT (XK(ZZ)); ", " INT (
YK(ZZ)); " TO " INT (XK(ZZ - 1)); ", " INT (YK(ZZ - 1)): NEXT ZZ: PRINT "
HPOINT "; INT (XK(1)); ", " INT (YK(1)); " TO "; XO; ", " YO
33290 RETURN
33400 REM find point near ideal point
33410 IF (XG - XM) = 0 THEN M = 9.999E17: GOTO 33430
33420 M = (YG - YM) / (XG - XM)

```

```

33430 XI = XM + SGN (XG - XM) * AA(II) / SQR (1 + M * M):YI = M * (XI -
XM) + YM: RETURN
33500 REM reverse caculation for joints
33510 PRINT "Going for reverse calculation"
33540 XG = XL(1):YG = YL(1):XM = XO:YM = YO:II = 1: GOSUB 33400:XK(1) = X
I:YK(1) = YI:XM = XK(3):YM = YK(3):D = (XM - XI) ^ 2 + (YM - YI) ^ 2

33550 K1 = (D + AA(2) * AA(2) - AA(3) * AA(3)) / (2 * AA(2)):K2 = K1 / SQR
(D): IF (XM - XI) = 0 THEN AL = PI / 2: GOTO 33570
33560 AL = ATN ((YM - YI) / (XM - XI)): IF XM < XI THEN AL = AL + PI
33570 IF K2 > 1 THEN K2 = .999: PRINT CHR$(7): PRINT "Point (" INT (XP
)," INT (YP))" is too far away !": IF C% < 3 THEN F% = 1: RETURN
33580 TH = AL + ATN (SQR (1 - K2 * K2) / K2):XL = XI + AA(2) * COS (TH
):YL = YI + AA(2) * SIN (TH)
33590 TH = AL - ATN (SQR (1 - K2 * K2) / K2):XI = XI + AA(2) * COS (TH
):YI = YI + AA(2) * SIN (TH): IF (XI - XK(2)) ^ 2 + (YI - YK(2)) ^
2 < (XL - XK(2)) ^ 2 + (YL - YK(2)) ^ 2 THEN XL = XI:YL = YI
33600 XK(2) = XL:YK(2) = YL: RETURN
35000 REM Up Solution for no obstacles
35005 PRINT "Seeking Up Solution"
35010 D2% = 0:L2%(D2%) = 3:X(3,D2%) = XO:Y(3,D2%) = YO:LE = L / .9:LD = SQR
((XP - XO) ^ 2 + (YP - YO) ^ 2): IF XP = XO THEN M = 9.999E17: GOTO
35030
35020 M = (YP - YO) / (XP - XO)
35030 OP = (LE - LD) / 2:NC = 1:DC = OP / SQR (1 + M * M):Y(1,D2%) = Y(O
,D2%) + NC * DC:X(1,D2%) = X(O,D2%) + M * (Y(O,D2%) - Y(1,D2%))
35035 B = Y(O,D2%) - NC * DC:A = X(O,D2%) + M * (Y(O,D2%) - B): IF (A - X
K(1)) ^ 2 + (B - YK(1)) ^ 2 < (X(1,D2%) - XK(1)) ^ 2 + (Y(1,D2%) - Y
K(1)) ^ 2 THEN X(1,D2%) = A:Y(1,D2%) = B:NC = - 1
35040 Y(2,D2%) = Y(3,D2%) + NC * DC:X(2,D2%) = X(3,D2%) + M * (Y(3,D2%) -
Y(2,D2%)):KK = 0: GOSUB 29500
35050 IF ABS (Q - L) < 10 THEN LH(KK) = Q: GOSUB 33000: RETURN
35060 LE = LE * L / Q: GOTO 35030
39990 REM error handling routine
40000 IF Z1 > 279 THEN Z1 = 279: RESUME
40010 IF Z1 < 0 THEN Z1 = 0: RESUME
40020 IF Z2 > 191 THEN Z2 = 191: RESUME
40030 IF Z2 < 0 THEN Z2 = 0: RESUME
55000 PI = 3.14159265:BB = 180 / PI: INPUT "DX = ";DX: INPUT "DY = ";DY: GOSU
10500: PRINT "SLOPE = "; INT (SP * BB) " deg": GOTO 55000
60000 REM clear output file
60030 F0$ = FI$ + ".JNT": ONERR GOTO 60050
60040 PRINT CHR$(4);"DELETE";F0$
60050 PRINT CHR$(4);"OPEN";F0$
60060 RETURN

```

]

]

```

1 REM *****
2 REM *
3 REM * OBSTACLE AVOIDANCE *
4 REM * PROGRAM *
5 REM *
6 REM * Anupam Bagchi *
7 REM * Dec 1986 *
8 REM *
9 REM *****
10 DIM CGX(20)
15 PI = 3.14159265:BB = 180 / PI
20 DIM XE(15),YE(15),XB(10),YB(10),XIZ(25),YIZ(25)
300 GOSUB 9500: GOSUB 8200
320 ONERR GOTO 500
330 PRINT DO$: GOSUB 1500:F0$ = FI$ + ".JNT": PRINT DO$: PRINT DO$;"READ
";F0$: ON KEZ GOTO 700,800,600,1000,900,850,1700
340 PRINT DO$;"CLOSE";F0$
500 IF PEEK (222) = 5 THEN 1400
510 IF XX(JJ) < 0 THEN XX(JJ) = 0: RESUME
520 IF XX(JJ) > 279 THEN XX(JJ) = 279: RESUME
530 IF YY(JJ) < 0 THEN YY(JJ) = 0: RESUME
540 IF YY(JJ) > 191 THEN YY(JJ) = 191: RESUME
590 PRINT CHR$ (7);: PRINT "Something is wrong !": PRINT : PRINT "Exiti
ng because of confusion.": PRINT : HTAB 10: PRINT "I am sorry for th
at.": PRINT : PRINT "Type RUN to start program again.": TEXT : END
600 REM superposed configurations
610 GOSUB 50200: GOSUB 8300
620 FOR KK = 1 TO 400: NEXT KK: FOR JJ = 1 TO NLZ + 1: INPUT XX(JJ),YY(J
J): NEXT JJ: HPLOT XX(1),YY(1): FOR JJ = 2 TO NLZ + 1: HPLOT TO XX(
JJ),YY(JJ): NEXT JJ
630 GOTO 620
700 REM list points on terminal
705 IZ = 1
710 FOR JJ = 1 TO NLZ + 1: INPUT XX(JJ),YY(JJ): NEXT JJ: PRINT : HTAB 8:
PRINT "Iteration number ";IZ: PRINT : PRINT XX(1),YY(1):CV = 0
720 FOR JJ = 2 TO NLZ + 1: PRINT XX(JJ),YY(JJ):CV = CV + SQR ((XX(JJ) -
XX(JJ - 1)) ^ 2 + (YY(JJ) - YY(JJ - 1)) ^ 2): NEXT JJ
730 HTAB 5: PRINT "LENGTH = ";CV:IZ = IZ + 1: GOTO 710
800 REM draw configuration one by one
810 GOSUB 50200: GOSUB 8300
820 FOR JJ = 1 TO NLZ + 1: INPUT XX(JJ),YY(JJ): NEXT JJ: HCOLOR= 3
830 GOSUB 840: GOSUB 8000: HCOLOR= 0: GOSUB 840: GOTO 820
840 HPLOT XX(1),YY(1): FOR JJ = 2 TO NLZ + 1: HPLOT TO XX(JJ),YY(JJ): NEXT
JJ: RETURN
850 REM plot endeffector path
860 GOSUB 50200: GOSUB 8300
870 FOR JJ = NLZ TO 0 STEP - 1: INPUT XX(JJ),YY(JJ): NEXT JJ
880 HPLOT XX(NLZ),YY(NLZ): GOTO 870
900 REM calculate,plot & display JOINT COORDINATES
910 FOR JJ = NLZ TO 0 STEP - 1: INPUT XX(JJ),YY(JJ): NEXT JJ
950 DX = XX(1) - XX(0):DY = YY(1) - YY(0): GOSUB 10500:VB = SP: IF ABS
VB) > PI THEN VB = VB - SGN (VB) * 2 * PI
955 PRINT INT (VB * BB * 10 + .5) / 10
960 FOR JJ = 2 TO NLZ:M1 = SP:DX = XX(JJ) - XX(JJ - 1):DY = YY(JJ) - YY
JJ - 1): GOSUB 10500: HTAB 7 * (JJ - 1):VB = SP - M1
970 IF ABS (VB) > PI THEN VB = VB - SGN (VB) * 2 * PI
980 PRINT SGN (VB) * INT ( ABS (VB) * BB * 10 + .5) / 10: NEXT JJ
990 GOTO 910

```

```

1000 REM draw joint angles
1010 PRINT DO$: PRINT : PRINT "Which joint angle do you want to plot ?":
PRINT : PRINT : HTAB 10: INPUT "ANGLE NUMBER = ";KZ
1012 PRINT : PRINT : HTAB 10: INPUT "MAGNIFICATION = ";MA: PRINT : PRINT
: HTAB 10: INPUT "STEP LENGTH = ";ICZ
1015 PRINT DO$;"READ";FO$
1020 HGR2 : HCOLOR= 3:IX = 0:MDX = 100: ONERR GOTO 1300
1030 HPLOT 0,191 TO 0,0 TO 279,0: FOR JJ = 10 TO 270 STEP 10: HPLT JJ,0
TO JJ,3: NEXT : FOR JJ = 10 TO 180 STEP 10: HPLT 0,JJ TO 3,JJ: NEXT

1035 HPLT 279,MDX TO 0,MDX:XIX = 0:YIX = MDX
1040 FOR JJ = NLX TO 0 STEP - 1: INPUT XX(JJ),YY(JJ): NEXT JJ
1050 IF KZ < 1 OR KZ > NLX THEN 1100
1060 IF KZ = 1 THEN 1090
1070 DX = XX(KZ - 1) - XX(KZ - 2):DY = YY(KZ - 1) - YY(KZ - 2): GOSUB 105
00:M1 = SP:DX = XX(KZ) - XX(KZ - 1):DY = YY(KZ) - YY(KZ - 1): GOSUB
10500:VB = SP - M1
1075 IF ABS (VB) > PI THEN VB = VB - SGN (VB) * 2 * PI
1080 XJZ = ICZ * IX:YJZ = VB * MA + MDX: HPLT XIZ,YIZ TO XJZ,YJZ:XIZ = X
JZ:YIZ = YJZ:IX = IX + 1: GOTO 1040
1090 DX = XX(KZ) - XX(KZ - 1):DY = YY(KZ) - YY(KZ - 1): GOSUB 10500:VB =
SP: IF ABS (VB) > PI THEN VB = VB - SGN (VB) * 2 * PI
1095 GOTO 1080
1100 DX = XX(1) - XX(0):DY = YY(1) - YY(0): GOSUB 10500: IF ABS (SP) > P
I THEN SP = SP - SGN (SP) * 2 * PI
1105 JC(1) = SP
1110 FOR CC = 2 TO NLX:M1 = SP:DX = XX(CC) - XX(CC - 1):DY = YY(CC) - YY
(CC - 1): GOSUB 10500:JC(CC) = SP - M1: IF ABS (JC(CC)) > PI THEN J
C(CC) = JC(CC) - SGN (JC(CC)) * 2 * PI
1115 NEXT CC
1120 XJZ = ICZ * IX: FOR CC = 1 TO NLX:YJZ = JC(CC) * MA + MDX: HPLT XIZ
,JM(CC) TO XJZ,YJZ:JM(CC) = YJZ: NEXT CC:XIZ = XJZ:IX = IX + 1: GOTO
1040

1300 IF PEEK (222) = 5 THEN 1400
1310 IF YJZ < 0 THEN YJZ = 0: RESUME
1320 IF YJZ > 191 THEN YJZ = 191: RESUME
1330 IF XJZ > 279 THEN XJZ = 279: RESUME
1340 IF XJZ < 0 THEN XJZ = 0: RESUME
1350 RESUME
1360 GOTO 590
1400 REM asks for a choice of replay
1410 SK = - 16336:X = PEEK (SK) + PEEK (SK)
1420 KK = RND (1) * 255: FOR I = 1 TO KK: NEXT I
1430 KE = PEEK ( - 16384): IF KE < 128 THEN 1410
1433 IF KE < > 155 THEN 1440
1435 POKE - 16368,0
1437 KE = PEEK ( - 16384): IF KE < 128 THEN 1437
1440 POKE - 16368,0: TEXT : HOME : VTAB 6:A$ = "Do you want to run prog
ram again ?": GOSUB 9000: PRINT : HTAB 15: PRINT "(Yes/No) ";
1450 GET KE$: IF KE$ = "Y" OR KE$ = "N" THEN 1470
1460 GOTO 1450
1470 IF KE$ = "N" THEN END
1480 GOTO 320
1500 REM asks for choice
1510 TEXT : HOME :A$ = "CHOOSE MODE OF DISPLAY": VTAB 4: GOSUB 9000: VTAB
7: PRINT "Press": PRINT : PRINT "1 to List Coordinates on screen
": PRINT : PRINT "2 to Draw configuration one by one"
1520 PRINT : PRINT "3 to Draw superposed configurations": PRINT : PRINT
"4 to Plot Joint Coordinates wrt time": PRINT "5 to List Joi
nt coordinates "

```



```

1525 PRINT : PRINT "6 to Plot End Effector path ": PRINT : PRINT "7
to Draw selected configurations ";
1530 GET KE$:KEZ = ASC (KE$): IF KEZ < 49 OR KEZ > 55 THEN 15301540KEZ =
KEZ - 48
1540 KEZ = KEZ - 48
1590 RETURN
1700 REM plot only selected configurations
1710 PRINT DO$: HOME : VTAB 5:A$ = "Enter the configuration numbers": GOSUB
9000: PRINT :A$ = "you want to plot.": GOSUB 9000: PRINT : PRINT : HTAB
5: INPUT "Number of configurations = ";NMZ
1720 FOR KK = 1 TO NMZ: PRINT : HTAB 2: PRINT "<";KK;"> Configuratio
n number = ";: INPUT " ";CGZ(KK): NEXT KK
1725 GOSUB 50200: GOSUB 8300
1730 PRINT DO$;"READ";FO$:IZ = 1
1740 FOR JJ = 1 TO NLZ + 1: INPUT XX(JJ),YY(JJ): NEXT JJ
1750 GOSUB 1900: IF SUZ = 1 THEN HPLLOT XX(1),YY(1): FOR JJ = 2 TO NLZ +
1: HPLLOT TO XX(JJ),YY(JJ): NEXT JJ
1760 IZ = IZ + 1: GOTO 1740
1900 REM tells whether present configuration needs to be plotted
1910 SUZ = 0:JJ = 0
1920 JJ = JJ + 1: IF JJ > NMZ THEN SUZ = 0: RETURN
1930 IF CGZ(JJ) = IZ THEN SUZ = 1: RETURN
1940 GOTO 1920
2100 REM read network from disk
2110 TEXT : HOME : VTAB 6: PRINT "Which file do you want to operate upon
?";
2120 VTAB 9: HTAB 10: INPUT "FILE NAME ==> ";FI$: IF LEN (FI$) = 0 THEN
2120
2130 IF LEN (FI$) > 4 AND ( RIGHT$ (FI$,4) = ".DAT" OR RIGHT$ (FI$,4) =
".NET" OR RIGHT$ (FI$,4) = ".PIC" OR RIGHT$ (FI$,4) = ".JNT") THEN
FI$ = LEFT$ (FI$, LEN (FI$) - 4)
2140 FO$ = FI$ + ".NET"
2150 DO$ = CHR$ (4): PRINT DO$: PRINT DO$;"READ";FO$
2160 INPUT XO,YO,ECZ: FOR JJ = 1 TO ECZ: INPUT XE(JJ),YE(JJ): NEXT JJ
2170 INPUT OCZ: FOR JJ = 1 TO OCZ: INPUT XB(JJ),YB(JJ),RSZ(JJ),RD(JJ),AN
(JJ): NEXT JJ: INPUT NLZ:L = 0: FOR JJ = 1 TO NLZ: INPUT AA(JJ):L =
L + AA(JJ): NEXT JJ
2180 INPUT EDZ: FOR JJ = 1 TO EDZ: INPUT ED$(JJ),TMZ(JJ),TNZ(JJ),HX(JJ),
HY(JJ),TX(JJ),TY(JJ),EX(JJ),EY(JJ): NEXT JJ
2190 PRINT DO$;"CLOSE";FO$: RETURN
8000 KEZ = PEEK ( - 16384): IF KEZ < 128 THEN 8000
8010 POKE - 16368,0: RETURN
8200 REM read picture from disk
8210 FO$ = FI$ + ".PIC"
8230 DO$ = CHR$ (4): PRINT DO$;"READ";FO$: INPUT KIZ: FOR KK = 1 TO KIZ:
INPUT XIX(KK),YIX(KK): NEXT KK: PRINT DO$;"CLOSE";FO$
8240 RETURN
8300 REM draw picture
8310 IF KIZ = 0 THEN RETURN
8320 FOR II = 1 TO KIZ: IF YIX(II) > 1000 THEN HPLLOT XIX(II),YIX(II) -
1000: GOTO 8340
8330 HPLLOT TO XIX(II),YIX(II)

```

```

8340 NEXT II
8350 RETURN
9000 B = INT (20 - ( LEN (A$) / 2)): IF B = < 0 THEN B = 1
9010 HTAB B: PRINT A$: RETURN
9500 REM read stored data
9510 TEXT : HOME : VTAB 6: PRINT "Which file do you want to operate upon
?";
9520 VTAB 9: HTAB 10: INPUT "FILE NAME ==> ";FI$: IF LEN (FI$) = 0 THEN
9520
9530 IF LEN (FI$) > 4 AND ( RIGHT$ (FI$,4) = ".DAT" OR RIGHT$ (FI$,4) =
".NET" OR RIGHT$ (FI$,4) = ".PIC" OR RIGHT$ (FI$,4) = ".JNT") THEN
FI$ = LEFT$ (FI$, LEN (FI$) - 4)
9540 FO$ = FI$ + ".DAT"
9550 DO$ = CHR$ (4): PRINT DO$: PRINT DO$;"READ";FO$: INPUT XO,YO,ECZ: FOR
KK = 1 TO ECZ: INPUT XE(KK),YE(KK): NEXT KK: INPUT OCZ: FOR KK = 1 TO
OCZ: INPUT XB(KK),YB(KK),RSZ(KK): NEXT KK
9560 INPUT NLZ:L = 0: FOR KK = 1 TO NLZ: INPUT AA(KK):L = L + AA(KK): NEXT
KK: PRINT DO$;"CLOSE";FO$
9570 RETURN
10500 REM calculates THETA from vector
10520 IF DX = 0 THEN SP = PI / 2: GOTO 10540
10530 SP = ATN (DY / DX)
10540 IF DX < 0 THEN SP = SP + PI
10550 RETURN
50000 REM print all information
50010 FOR II = 1 TO OCZ: PRINT "XB(";II;") = ";XB(II);" YB(";II;") = ";Y
B(II): NEXT II
50020 GET KE$: FOR II = 1 TO OCZ: PRINT "RSZ(";II;") = ";RSZ(II): NEXT I
I: GET KE$: FOR II = 1 TO EDZ: PRINT "ED$(";II;") = ";ED$(II);" TMZ(
";II;") = ";TMZ(II);" TNZ(";II;") = ";TNZ(II): PRINT "HX(";II;") = "
;HX(II);" HY(";II;") = ";HY(II)
50030 PRINT "TX(";II;") = ";TX(II);" TY(";II;") = ";TY(II): PRINT "EX(";
II;") = ";EX(II);" EY(";II;") = ";EY(II): NEXT II
50040 RETURN
50100 REM print input data
50110 TEXT : HOME : VTAB 5: HTAB 5: PRINT "Starting point : (";XO;",";YO
;")": PRINT : HTAB 7: PRINT "Number of end points = ";ECZ: FOR JJ =
1 TO ECZ: HTAB 7: PRINT "End point ";JJ;" at (";XE(JJ);",";YE(JJ);")
": NEXT JJ
50120 PRINT : HTAB 7: PRINT "Number of obstacles = ";OCZ: FOR JJ = 1 TO
OCZ: HTAB 3: PRINT "Obstacle ";JJ;" at (";XB(JJ);",";YB(JJ);") with
radius ";RSZ(JJ): NEXT JJ
50130 IF NLZ > 0 THEN PRINT : HTAB 7: PRINT "Number of links = ";NLZ:L =
0: FOR JJ = 1 TO NLZ: HTAB 5: PRINT "Length of link ";JJ;" = ";AA(JJ
):L = L + AA(JJ): NEXT JJ: HTAB 9: PRINT "Total length = ";L: RETURN
50200 REM plot the points
50210 HGR2 : HCOLOR= 3: GOSUB 50300
50220 FOR JJ = 1 TO OCZ: GOSUB 50340: NEXT JJ
50290 RETURN
50300 HPLLOT 0,191 TO 0,0 TO 279,0
50310 HPLLOT 273,3 TO 279,10: HPLLOT 279,3 TO 273,10: HPLLOT 3,184 TO 7,188
: HPLLOT 4,191 TO 9,184
50320 FOR JJ = 10 TO 270 STEP 10: HPLLOT JJ,0 TO JJ,3: NEXT : FOR JJ = 10
TO 180 STEP 10: HPLLOT 0,JJ TO 3,JJ: NEXT
50330 RETURN
50340 FOR KK = 0 TO 360 STEP 10:X1 = XB(JJ) + RSZ(JJ) * COS (KK / BB):Y
1 = YB(JJ) + RSZ(JJ) * SIN (KK / BB): HPLLOT X1,Y1: NEXT KK: RETURN

```


ME-1987-M-BAG-TRA .